

**ROTEAMENTO E GERÊNCIA DE MOBILIDADE
NA INTERNET DAS COISAS**

BRUNO PEREIRA DOS SANTOS

**ROTEAMENTO E GERÊNCIA DE MOBILIDADE
NA INTERNET DAS COISAS**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: ANTONIO ALFREDO FERREIRA LOUREIRO
COORIENTADOR: LUIZ FILIPE MENEZES VIEIRA

Belo Horizonte

Agosto de 2019

BRUNO PEREIRA DOS SANTOS

ROUTING AND MOBILITY MANAGEMENT IN
THE INTERNET OF THINGS

Thesis presented to the Graduate Program
in Computer Science of the Federal Univer-
sity of Minas Gerais in partial fulfillment of
the requirements for the degree of Doctor
in Computer Science.

ADVISOR: ANTONIO ALFREDO FERREIRA LOUREIRO
CO-ADVISOR: LUIZ FILIPE MENEZES VIEIRA

Belo Horizonte

August 2019

© 2019, Bruno Pereira dos Santos.
Todos os direitos reservados.

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg
Lucas Cruz – CRB 6^a Reg. n^o 819.

Santos, Bruno Pereira dos

S237r Routing and Mobility Management in the Internet of
Things / Bruno Pereira dos Santos. — Belo Horizonte,
2019

xxxiii, 128 f. . il.; 29cm

Tese (doutorado) — Universidade Federal de Minas
Gerais – Departamento de Ciência da Computação.

Orientador: Antonio Alfredo Ferreira Loureiro

Coorientador: Luiz Filipe Menezes Vieira

1. Computação – Teses. 2. Redes de Sensores Sem Fio
– Teses. 3. Internet das Coisas – Teses. 4. Roteamento
(Administração de redes de computadores) – Teses.

I. Orientador. II. Coorientador. III. Título.

CDU 519.6*22(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE POS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO


FOLHA DE APROVAÇÃO

Routing and Mobility Management in Internet of Things

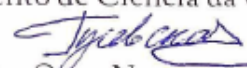
BRUNO PEREIRA DOS SANTOS


Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:


PROF. ANTONIO ALFREDO FERREIRA LOUREIRO - Orientador
Departamento de Ciência da Computação - UFMG


PROF. LUIZ FILIPE MENEZES VIEIRA - Coorientador
Departamento de Ciência da Computação - UFMG


PROF. MARCOS AUGUSTO MENEZES VIEIRA
Departamento de Ciência da Computação - UFMG


PROFA. OLGA NIKOLAEVNA GOUSSEVSKAIA
Departamento de Ciência da Computação - UFMG


PROF. MARKUS ENDLER
Departamento de Informática - PUC-Rio


PROFA. RAQUEL APARECIDA DE FREITAS MINI
Departamento de Ciência da Computação - PUC-MG


PROF. ANDRÉ LUIZ LINS DE AQUINO
Instituto de Computação - UFAL

Belo Horizonte, 9 de Agosto de 2019.

Acknowledgments

My warm thanks to everyone who supported me and contributed in different ways to the development of this work. I would first like to thank my advisor Antonio Alfredo Ferreira Loureiro for his teaching, knowledge, and patience during all doctoral process. Also, I thank my co-advisor Luiz Filipe Menezes Vieira for all support and enlightening talks. Their guidance and expertise were fundamental in the formulating of the research topic and methodology.

I would like to thank the faculty UFMG and UESC and all supporting staff. The professors Olga Nikolaevna Goussevskaia, Marcos Augusto M. Vieira, Pedro O.S. Vaz de Melo for their willingness to support me, clarify doubts and give suggestions.

Also, I would like to thank my colleagues for helping me during the Ph.D. process. Especially, Paulo Henrique, Clayson Celes, Otávio Augusto, Evellyn Cavalcante, and Maria Gabriela, WISEMAP/WINET labs, and Qualis Mission study group.

I also want to thank my family and dear friends for affection, encouragement, and friendship. I hid the names of all people that I would like to thank since this work without all the people in my life would not be possible. I am thankful for all.

I would like to thank FAPEMIG, CNPq, and CAPES for their financial support for this research.

*“Que a força do medo que tenho
Não me impeça de ver o que anseio
Que a morte de tudo em que acredito
Não me tape os ouvidos e a boca
Porque metade de mim é o que eu grito
Mas a outra metade é silêncio. ”*
(Oswaldo Montenegro)

Resumo

No futuro da Internet das Coisas (Internet of Things (IoT)), os objetos do dia-a-dia irão, provavelmente, ser conectados à Internet e, deste fato, emergem diversas oportunidades tanto no âmbito de pesquisa quanto comercial. Atualmente, a IoT interconecta predominantemente objetos estáticos anexados em alguma infraestrutura física. No entanto, a mobilidade é um fator presente em nossas rotinas diárias, como consequência disto, os objetos (“things”) serão transportados ou poderão se mover em um ambiente cibernético conectado. A este ambiente dá-se o nome de Internet das Coisas Móveis (Internet of Mobile Things (IoMT)) e, ao se moverem, eventualmente os dispositivos poderão criar laços sociais surgindo então a Internet das Coisas Sociais (Social Internet of Things (SIoT)). A mobilidade é uma questão de pesquisa no âmbito da IoT, a qual desempenha papel importante no desenho, implementação, bem como a “ubiquitificação” destes dispositivos em nossas vidas. Apesar da importância da mobilidade no contexto das redes de computadores, gerenciá-la adequadamente, especialmente no contexto da IoT, ainda é uma questão de pesquisa em aberto.

Nesta tese, investigamos o roteamento e a gestão da mobilidade no contexto da IoT. Avaliamos quais são as boas práticas para estas soluções, bem como estudamos a mobilidade em IoT em diferentes contextos. Soluções já foram propostas para dar suporte à IoT onde os dispositivos são estáticos. Contudo, pouco se tem feito quando a mobilidade é presente, o que motiva esta tese. Deste modo, damos um passo além no estudo sobre o roteamento ciente da mobilidade para IoT ao explorar os blocos básicos de construção para manipular mobilidade a nível das entidades (comunicação móvel, dispositivo móveis e *software*) bem como as operações básicas para gerência da comunicação (detecção de mobilidade, processo de *handover* e gerência da mobilidade). Argumentamos, também, que para manter a comunicação sob eventos de mobilidade, o roteamento apresenta um papel fundamental no processo. Portanto, as nossas contribuições situam-se nesse contexto. Neste sentido, propusemos soluções de roteamento chamadas Dribble, Matrix e Mobile Matrix que são alternativas para respectivamente a detecção de mobilidade de entidades, o roteamento estático e o roteamento sob eventos

de mobilidade. Estas propostas apresentam melhorias, quando comparados às soluções existentes na literatura, quanto a eficiência no uso de recursos de memória e energia, confiabilidade, tolerância a falhas, bem como no gerenciamento da comunicação com entidades móveis.

Palavras-chave: Internet das Coisas, Mobilidade, Endereçamento Hierárquico, Protocolo de Roteamento.

Abstract

In the future Internet of Things (IoT), everyday objects will probably be connected to the Internet, leading to research and commercial opportunities. Current IoT interconnects mainly static devices attached to some physical infrastructure. However, mobility is present in our daily life and, thus, naturally objects can move around forming so-called IoMT and create social ties composing the SIoT in the cyber-physical space. Mobility is a fundamental research field in IoT context, being critical to its design and implementation, as well as the “ubiquitification” of those devices in our life. This raises the question of whether existing ready-to-use networking protocols are enough and suitable to cover social (Social IoT) and mobile (IoMT) demands. Despite the importance of mobility for IoT, the field is not thoroughly studied and understood regarding properly handle mobility, especially concerning constrained devices typically used in IoT.

In this thesis, we have investigated new mechanisms of routing and mobility management for IoT. Our initial motivation for this research was that many solutions had been proposed to support static IoT. However, little attention has been given to the mobility factor in such a context. Here, we take a step forward on the study of mobility solutions in terms of routing and mobility management for IoT by exploring the building blocks to handle mobility properly (mobile communication, mobile devices, and software) as well as to the basic operations to manage mobility communication (mobility detection, handover process, and mobility management). In such a scenario, we argue that routing is an IoT enabler playing a key role in handling mobility. In this thesis, we have proposed routing solutions named Dribble, Matrix, and Mobile Matrix that are alternatives to deal with IoMT demands for mobility detection, routing for static devices and routing under mobile environment. Our results advance the state-of-the-art of the IoMT in terms of efficiency in energy and memory usage, reliability, efficiency, fault tolerance, as well as mobility management.

Palavras-chave: IoT, Mobility, Hierarchical Address, Routing protocol.

List of Figures

1.1	Requirements for a Mobile IoT.	3
1.2	Building block operations to handle mobility.	4
2.1	Interest for IoT and Wireless Sensor Networks (WSN) terms on Google platform.	8
2.2	Hypercycle of July 2018. The IoT appears in the <i>Peak of Inflated Expectations</i> and IoT's Platforms appears as a <i>Technology Trigger</i> . Source: [45].	8
2.3	IoT as an Internet extension. Source: [117].	9
2.4	IoT's basic building blocks: from technology to human value.	10
2.5	Smart devices interconnection with Internet example.	12
2.6	Example of the Internet Protocol version 6 (IPv6) packet header with and without compression.	13
2.7	Classification of IoT wireless communication technologies by spatial coverage.	14
4.1	(a) the Multipoint-to-Point (M2P) routing structure that creates routes from nodes towards the root. (b) presents the Point-to-Multipoint (P2M) routing structure typically created through control packets sent over (a) structure. (c) the three main data traffic patterns over the routing structure constructed on (a) and (b).	30
4.2	Routing Protocol for Low-Power and Lossy Networks (RPL) control packets.	32
4.3	Timer schemes' basic trade-off faced.	33
4.4	Three main timer schemes commonly used in IoT routing protocols.	34
4.5	Dribble flowchart diagram.	35
4.6	(a) a trace of raw GPS points forming a trajectory of an entity. (b) extracted Stay Location from a trajectory (a). (b) Stay location history.	39
4.7	Mobility metrics for each entity.	42
4.8	Confusion matrix.	43
4.9	The trade-off between control overhead and disconnection time.	44

4.10	The control flow overhearing and the average time in a floating state. . . .	44
4.11	The delivery rate for different evaluated scenarios.	46
5.1	Matrix protocol's architecture.	52
5.2	Multihop Host Configuration for 6LoWPAN (MHCL): simplified IPtree example: 8-bit address space at the root and 6.25% reserved for future/delayed connections.	55
5.3	RCtree example: before and after two links change in the collection tree. .	58
5.4	Alternative top-down routing upon Ctree change.	61
5.5	Routing table usage Complementary Cumulative Distribution Function (CCDF). The routing table size was setted to 20 entries.	66
5.6	Code and memory footprint in bytes.	67
5.7	Number of control packets.	68
5.8	Top-down routing success rate.	69
5.9	Any-to-any routing success rate.	71
5.10	Response success rate.	72
6.1	μ Matrix integrated into the network stack.	79
6.2	μ Matrix protocol's architecture.	81
6.3	Routing structures: Ctree, IPtree, and RCtree.	82
6.4	Simplified hierarchical address assignment with 8-bit available address space and 6.25% of addresses reserved for delayed nodes. Inside the nodes, its label and IP assigned, the % next to the nodes express the approximate sub-tree size. Thick downwards arrows indicate the available IP range fairly distributed.	83
6.5	Reverse Trickle Timer operation with μ Matrix.	85
6.6	Mobile Engine state machine.	86
6.7	Mobile engine operation after mobility events.	87
6.8	μ Matrix's preserves locality when it updates the routing table under mobility events. <i>Mtables</i> above LCA do not need updates.	88
6.9	CDF of routing table usage. For μ Matrix <i>Mtable + IPchildren</i> , for RPL only downwards routing table. The maximum table size is 20.	98
6.10	The number of control packets.	99
6.11	Bottom-up routing success rate.	101
6.12	Top-down routing success rate. The transparent bar represents inevitable losses.	102

6.13	The trade-off between control message overhead and the successful delivery rate.	103
6.14	The trade-off between the delivery rate with acknowledgment and RTT. . .	104
7.1	Lisp packet flow. (EID: endpoint identifier; ETR: egress tunnel router; ITR: ingress tunnel router; PxTR proxy tunnel router; RLOC: routing locator). Source: [115].	112

List of Tables

4.1	A non-exhaustive list of mobility metrics, a short description and its classification.	37
4.2	Default simulation parameters	40
4.3	Mobility models parameters	41
4.4	Model parameters and classification report	42
5.1	Comparison between related protocols for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN).	51
5.2	Simulation parameters.	65
5.3	Faulty network scenarios.	65
6.1	Routing protocol properties (RevTT – Reverse Trickle Timer; RTL – Traver Trickle Timer Like; T – Trickle; P – Periodic; HL – Home Location Assumption; SN – Static Nodes.)	76
6.2	GRM parameters.	95
6.3	CRWP parameters.	97
6.4	CRWP mobility metrics.	97
6.5	Simulation parameters.	99

List of Acronyms

- 3GPP** Third Generation Partnership Project. 15
- 6LoWPAN** IPv6 over Low-Power Wireless Personal Area Networks. xxi, 12–14, 23, 47–49, 51, 53, 57, 59, 64, 71–76, 78, 85, 87
- AODV** Ad hoc On-Demand Distance Vector. 50, 78, 99–105
- BLE** Bluetooth Low Energy. 14, 17
- Bluetooth SIG** Bluetooth Special Interest Group. 17
- BT** Bluetooth. 17
- CBFR** Counting Bloom Filter Routing. 48
- CCDF** Complementary Cumulative Distribution Function. xviii, 66
- CDF** Cumulative Distribution Functions. xviii, 98, 100
- CGR** Centrality-based Green Routing for Low-Power and Lossy Networks. 51
- Co-RPL** Corona RPL. 76, 77
- CODU** Contact duration. 37
- CRWP** Cyclical Random Waypoint. 27, 41, 75, 96, 97, 100, 101, 104, 105
- CTP** Collection Tree Protocol. 24, 25, 47–51, 53–55, 57, 59, 60, 64, 65, 67, 68, 73, 77, 79–81, 85, 90, 93
- DAG** Directed Acyclic Graph. 31
- DAO** DODAG Destination Advertisement Object. 31, 32, 45
- DAO-Ack** DODAG Destination Advertisement Acknowledgement. 31

DIO DODAG Information Object. 31, 32, 40, 45

DIS DODAG Information Solicitation. 31, 40

DMMLIS Distributed Mobility Management scheme in Locator/Identifier Separation networks. 112

DODAG Destination Oriented Directed Acyclic Graph. 24, 31, 32, 40, 45

DRM DAG-based Multipath Routing. 76, 77

DSR Dynamic Source Routing Protocol. 50

EDGEF Encounter regularity. 37

EXI Efficient XML Interchange. 11

GRM Group Regularity Mobility Model. 26, 27, 41, 95, 96, 100–104

HF High Frequency. 16

HMIPv6 Hierarchical Mobile IPv6. 78

INCO Inter-contact time. 37

IoMT Internet of Mobile Things. xiii, xv, 1, 3, 6, 10, 19–21, 73–76, 78, 79, 94, 100, 104–106

IoT Internet of Things. xiii, xv, xvii, 1–12, 14–16, 18–21, 23–27, 29–38, 46, 47, 51, 73–76, 78, 79, 94, 99, 100, 104, 105, 107, 108, 111–114

IP Internet Protocol. 12, 23, 47, 59, 111

IPv6 Internet Protocol version 6. xvii, 12–14, 21, 108

LCA Lowest Common Ancestor. 53, 59, 61, 89

LF Low Frequency. 16

LISP Locator/Identifier Separation Protocol. 111

LPWAN Low Power Wide Area Network. 14, 15

M2P Multipoint-to-Point. xvii, 30, 31, 40

MA Mobile Agent. 113

ME-RPL Mobility Enhanced RPL. 76, 77

MHCL Multihop Host Configuration for 6LoWPAN. xviii, 52–55, 64–67, 70

MIMO Multiple Input Multiple Output. 15, 114

MIPv6 Mobile IPv6. 72, 78

MLP Multi-Layer Perceptron. 37, 42

MMRPL Mobility Management RPL. 76, 77, 99, 100, 102–105

mRPL Mobile RPL. 76, 77

MRPL-V Mobile RPL to Vehicular Networks. 76, 77

MTU Maximum Transmission Unit. 13

NB-IoT Narrowband IoT. 15, 16

NFC Near-Field Communication. 14, 16, 17

OIoT Opportunistic IoT. 20

ORPL Opportunistic RPL. 48

P2M Point-to-Multipoint. xvii, 30, 31, 40

P2P Point-to-Point. 30, 31, 40

PRR Packet Reception Rate. 102, 103, 105

RADG Radius of gyration. 37

RDF Resource Description Framework. 11

RevTT Reverse Trickle Timer. xviii, 74, 75, 85–89, 101, 102, 108

RevTT Reverse Trickle Timer. 30, 34, 43, 45, 46

RFID Radio-Frequency IDentification. 7, 16

RPL Routing Protocol for Low-Power and Lossy Networks. xvii, xviii, 24, 25, 31–34, 40, 45, 47–49, 51, 53, 57, 59, 64, 66–70, 73, 76–81, 84, 85, 90, 93, 98–105, 108

RTT Round Trip Time. xix, 104

RWP Random Waypoint. 27, 96

SIoT Social Internet of Things. xiii, xv, 1, 3, 6, 10, 19–21, 73–76, 78, 79, 94, 100, 104–106, 112

SL Stay Location. 39

SLH Stay Location History. 39

SWIM Small World in Motion. 26, 95

TD Travel Distance. 41, 43

THL Time Has Lived. 83, 89, 90

TOPO Topological overlap. 37

TRVD Travel distance. 37

TRVT Travel time. 37

TT Trickle Timer. 29, 33, 34, 36, 43, 45, 46, 108

TTL Time To Lived. 83, 90

UAV Unmanned Aerial Vehicles. 18

UDG Unit Disk Graph. 62

UHF Ultra High Frequency. 16

VIST Visit time. 37

VT Visit Time. 41, 43

WLAN Wireless Local Area Network. 16

WMAN Wireless Metropolitan Area Networks. 14

WOL Web Ontology Language. 11

WPAN Wireless Personal Area Network. 16, 17

WSN Wireless Sensor Networks. xvii, 7, 8, 76, 113

XCTP eXtend Collection Tree Protocol. 25, 31, 48–51, 64, 66–68, 70, 76–78

List of Algorithms

1	Matrix: Stabilization timer	54
2	Matrix: Aggregation timer (non-root nodes)	56
3	Matrix: Aggregation timer (root)	56
4	Matrix: IPv6 address distribution	57

Contents

Acknowledgments	ix
Resumo	xiii
Abstract	xv
List of Figures	xvii
List of Tables	xxi
List of Acronyms	xxiii
List of Algorithms	xxvii
1 Introduction	1
1.1 Contextualization	1
1.2 Relevance	3
1.3 Goals	4
1.4 Contribution	5
1.5 Thesis Organization	5
2 Internet of Things Overview	7
2.1 Internet of Things History Perspective	7
2.2 What is and Why The Internet of Things?	9
2.3 IoT's Basic Building Blocks	10
2.4 Internet Integration	12
2.5 Communication Technologies	14
2.6 Mobility Meets The Internet of Things	18
2.6.1 New IoT Paradigms and Requirements	19
2.7 Concluding Remarks	21

3	Model and Definition of Problems	23
3.1	Definition of Problems	23
3.2	Models	25
3.2.1	Entities	26
3.2.2	Mobility and Dynamic Model	26
4	Mobility Detection	29
4.1	Contextualization	29
4.2	IoT Routing in a Nutshell	30
4.3	Related Work and Problem Statement	32
4.4	Dribble Design	35
4.4.1	Entities	36
4.4.2	Extraction of Mobility Metrics	36
4.4.3	Learn-based Model	37
4.4.4	Timer Scheme Matching	38
4.4.5	Motion Event	38
4.5	Evaluation	39
4.5.1	Modeling the Entities Mobility	41
4.5.2	Measuring Mobility	41
4.5.3	The Neural Network	42
4.5.4	Assigning Timer Schemes to Mobility Patterns	43
4.5.5	Simulation Results	43
4.6	Concluding Remarks	46
5	An Alternative Routing Protocol for the Static Internet of Things	47
5.1	Contextualization	47
5.2	Related Work	50
5.3	Design Overview	51
5.3.1	MHCL: Multihop Host Configuration for 6LoWPAN	53
5.3.2	Control Plane: Distributed Tree Structures	57
5.3.3	Data Plane: Any-to-Any Routing	59
5.3.4	Fault Tolerance and Network Dynamics	60
5.3.5	Alternative Routing: Geometric Rationale	62
5.4	Complexity Analysis	62
5.5	Evaluation	64
5.5.1	Simulation Setup	64
5.5.2	Results	66

5.6	Concluding Remarks	71
6	An Alternative Routing Protocol for the Mobile Internet of Things	73
6.1	Contextualization	73
6.2	Background and Related Work	76
6.3	Design Overview	78
6.3.1	Mobile Matrix Architecture	80
6.3.2	Control Plane: Routing Engine	80
6.3.3	Control Plane: Mobile Engine	84
6.3.4	Mobile Matrix Data Plane: Any-to-Any Routing	90
6.4	Complexity Analysis	90
6.4.1	Memory Footprint	91
6.4.2	Control Message Overhead	92
6.4.3	Routing Path Distortion	93
6.5	Mobility Modelling	94
6.5.1	Human Mobility Model	95
6.5.2	Non-human Mobility Model	96
6.6	Evaluation	99
6.6.1	Simulation Setup	99
6.6.2	Results	100
6.7	Concluding Remarks	105
7	Conclusions and Future work	107
7.1	Conclusions	107
7.1.1	Mobility Detection	107
7.1.2	Handover Process and Mobility Management	108
7.2	Publications	108
7.3	Open Problems and Future Work	111
7.3.1	Inter-domain Routing with Mobile Matrix	111
7.3.2	Social IoT: A Mobility Model	112
7.3.3	Mobile Agents and IoT	113
7.3.4	IoT on 5G context	114
	Bibliography	115

Chapter 1

Introduction

1.1 Contextualization

The explosive growing number of everyday objects connected to the Internet emerges as a result of rapid progress and adoption of embedded systems, wired and wireless communication, and mainly the use of devices able to sense and move in the environment. The set of objects or just “things”¹ placed within the cyber-physical environment, where they can sense, interact with one each other and with humans is named the Internet of Things (IoT) [125]. Also, mobility is a major factor present in our daily life and, thus, “things” can move around forming the so-called Internet of Mobile Things (IoMT) [86], where devices can move in the cyber-physical space. In the future, the Social Internet of Things (SIoT) will be a reality where the smart devices do social ties among themselves as well as with humans in the social cyber-physical-social environment [1]. Those new technological artifacts have the potential to change the Internet as we see nowadays and how entities (machines and humans) interact with one each other. Indeed, in past years, the IoT has been attracted attention because of its potential for new applications towards smart and ubiquitous environments, academic research, and industrial and commercial acceptance.

Shortly, the IoT is an extension of the actual Internet, such extension proportionate *everyday objects* to be connected to the Internet. These objects must have at least the capacity for computation and communication. By connecting them to the global system of interconnected computer network, we will be able to control them remotely. Moreover, it will be possible for the “things” to act as service users or providers. However, these new facilities and possibilities come with a wide range of academic and

¹We use everyday objects, smart objects or devices, objects, devices, and “things” interchangeably, unless explicitly explained.

industrial challenges, that vary from technical ones (e.g., protocols standardization, device constraints) to the social ones (e.g., privacy issues and security threats resolutions).

The concept of computer networks has been changed over the years. With the introduction of IoT, the concept will change again. Tanenbaum [121] said that early computer network was a “collection of autonomous computers interconnected by a *single technology*”. Such a technology may be wired or wireless. For Peterson [98], “the most important characteristic of computers networks is its generality” meaning that computer networks are built with general-purpose hardware and application (e.g., a phone call, TV signals, or even files transmission). Kurose and Ross [67] argue that previous concepts (i.e., before 2012) of computer networks are outdated and they do not reflect the large amount of non-traditional devices and link technologies employed in the “new” networks.

Smart objects play a fundamental role in the evolution of the computer network concept, especially regarding the tremendous amount of heterogeneous devices in the network and highly mobile dynamic requirements. Powered by computational and communication resources, the objects may have the ability of sensing and motion which transform their functionality. In this sense, nowadays not only conventional computers are connected to the Internet, but a large sort of heterogeneous devices (e.g., TVs, laptops, automobiles, smart-phones, game consoles, web-cams, so on and so forth) with different specifications, requirements, and functionalities. In this new context, the device plurality is evident and is increasing. Recent forecasts [90] indicate that 30 billions of devices will be online on the Internet by 2020. Upon using the device’s resources, it will be possible to detect the object context, control it, exchange information, interactions in different perspectives (e.g., human-human, human-objects, objects-objects), and access and provide services. At the same time, potential new applications (e.g., for smart cities, health-care, smart home, and others) and challenges (e.g., regulation, security, standards) emerge. It is important to highlight that standardization of technologies plays a crucial role in the success of IoT, which will increase the heterogeneity of devices connected to the Internet, making the IoT a reality.

The mobility is present in everyday life. It makes life easier and turns IoT applications more flexible. The usage of many devices for IoT can benefit from it, as is the case of today’s adoption of smartphones and tablets. By extending IoT to handle mobility, IoT becomes even more ubiquitous. By supporting mobility, it is expected that smart objects to be transported or move themselves during normal usage, and this fact does not inhibit its normal operation and communication exchanges.

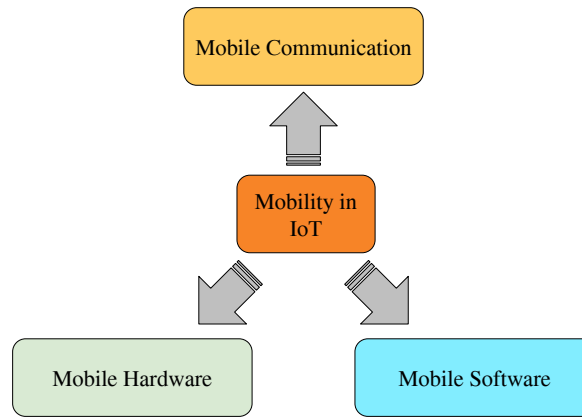


Figure 1.1: Requirements for a Mobile IoT.

1.2 Relevance

Many solutions have been studied and proposed to support static IoT. Also, standards were already done in the same context. However, handle mobility suitably remains an open issue in IoT [87]. The mobility factor will allow the flowering of new IoT paradigms, applications, and ubiquity interaction between humans and computer devices. Figure 1.1 shows the requirements to achieve a mobile IoT, in it, three main building blocks are highlighted: mobile communication, mobile hardware, and mobile software.

Issues regarding *Mobile Communication* usually encompass ad hoc networks as well as infra-structured networks. Thus, every network stack layer plays an essential role in mobile communication. *Mobile Hardware* comprises of mobile devices able to move by themselves or being attached to a mobile entity. The mobile pattern of the devices and humans can profoundly influence in mobile communication. Thus, this understatement is also an issue [8]. *Mobile Software* deals with the characteristics and requirements of mobile applications. Thus, for instance, an application may require specific types of communication patterns (or even delivery requirements) to provide its practical value to humans. These characteristics and requirements usually pose more challenges to the mobile IoT software development and designs.

This thesis focuses on the basic building blocks of mobile IoT. Therefore, any new mobile-based IoT application or paradigms may benefit from the results presented here. For instance, this study may help to make SIoT's [9] application a reality. Furthermore, this thesis advances current state-of-the-art by proposing mobility-aware solutions that can leverage IoT towards new mobile-based paradigms (such as IoMT and SIoT). These paradigms were predicted years ago when most of the IoT solutions were designed for static situations while mobile scenarios solutions were poorly considered. New IoT

mobile-based paradigms already have notoriety in a wide range of domains (e.g., smart home and cities, healthcare, smart automobiles) [5]. Agnihotri and Ramkumar [2] and Nahrstedt et al. [86] survey a bulk of applications and challenges in the mobile IoT context, and show the relevance of this thesis study and applicability.

1.3 Goals

Our focus is to propose advances in mobility-aware solutions for IoT, especially mobile communication solutions. In this context, the performance improvement and real adoption of mobile IoT’s applications are heavily influenced by the full mobility awareness of the network stack. From the lens of the network stack, we argue that the routing layer plays a fundamental role to the proper functioning, scalability, and identification of objects, especially regarding the highly dynamic network of “things”.

For these reasons, the scope of this thesis is the routing layer, but we are not limited to it. Indeed, we aim to investigate three basic operations to handle mobility in IoT: Mobility Detection, Handover, and Mobility Management processes. Figure 1.2 highlights those operations and their relationship. Such operations can be explored through different layers in the network stack, and can be influenced by endogenous and exogenous factors, such as the device mobility patterns and human intervention, respectively.

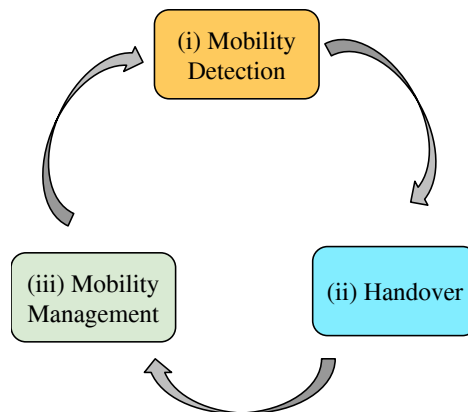


Figure 1.2: Building block operations to handle mobility.

Mobility Detection is the task of figuring out when a mobile entity is moving, specifically in our study within the network. The handover occurs when the entity changes its actual attachment point to another one. The Mobility Management is the task of keeping track of the current location of a mobile entity. These basic operations raise many specific challenges in IoT, for which solutions must be proposed: (i) one

specific challenge in Mobility Detection is that frequently IoT devices have only basic components (e.g., processing and communication units), thus how can the network stack protocols figure out that the entity is moving? Note that protocols may not have dedicated location hardware to query this information. (ii) regarding mobile handover and management, from the point-view of routing, how to promote transparent mobility and hide it from changes in the transport and application layers (e.g., address changes)?

1.4 Contribution

As mentioned, this thesis aims to improve the performance and support of mobile solutions and applications for IoT. We started by proposing a mobility detection mechanism named Dribble, which uses a leaning model to predict the devices' mobility pattern and then assigns a proper motion probing scheme accordingly. Then, we proposed a novel routing protocol and addressing scheme for static devices named *Matrix*. In future IoT, mobility will be omnipresent, thus we also proposed *Mobile Matrix* taking a step in such direction. *Mobile Matrix* is a novel mobility-aware routing protocol that provides an alternative to mobility detection, handover, and mobility management.

We summarize the main contributions of this thesis as follows.

1. Proposal of a machine learning based algorithm named Dribble that helps the mobility detection process for IoT devices.
2. Proposal of a new routing protocol named Matrix for static IoT. We designed the protocol to provide any-to-any routing, memory efficiency, fault tolerance, IPv6 enabled, and hardware independence;
3. A general overview of mobility, especially regarding the IoT. It helps to understand the state-of-the-art and issues concerning mobility in IoT;
4. Mobile Matrix as an extension of Matrix to support mobility devices. Mobile Matrix inherits all features from Matrix, but it does a step further by being able to manage mobility devices. To do that, it performs all building block operations presented in Figure 1.2. Also, Mobile Matrix was designed taking into the account cyclical mobility pattern behavior commonly performed by human and non-human entities.

1.5 Thesis Organization

We organize the rest of this thesis as follows.

- Chapter 2 gives an overview of IoT which is the background of this thesis. We describe the IoT history perspective and why to study this subject. Also, we describe the IoT's basic building blocks, as well as the mobility in the IoT context and new mobile-based IoT paradigms.
- Chapter 3 gives an overview of models and problems presented in the following chapters. The problems are contextualized first and then the models regarding topology dynamics are highlighted.
- Chapter 4 presents a learn-based timer scheme selector to support mobility detection of IoT devices. It uses a learning model to automatically predict devices' mobility pattern and then it assigns a proper motion probing timer schemes suitably.
- Chapter 5 presents an alternative routing protocol for IoT which we named Matrix Protocol. The Matrix can provide any-to-any routing under static and temporary faulty topologies. We show its architecture, formal analysis, and evaluation through simulations.
- Chapter 6 presents an alternative routing protocol for mobile IoT scenarios, which we named Mobile Matrix Protocol (μ Matrix). μ Matrix holds the same features as Matrix, but it can execute under mobile scenarios. We present its architecture, formal analysis, and evaluation through simulations. In our simulation experiments, μ Matrix was tested under human and non-human mobility patterns, showing that μ Matrix has the potential to provide support to SIoT and IoMT paradigms.
- Chapter 7 concludes this thesis by summarizing the main results achieved as well as pointing out the future work and open problems.

Chapter 2

Internet of Things Overview

2.1 Internet of Things History Perspective

The Internet of Things term was likely coined by Kevin Ashton in 1999 on one of his talks [7]. He argues that “Radio-Frequency IDentification (RFID) (we return to this technology later) and sensor technology will enable computers to observe, identify and understand the world – without the limitations of human-entered data”. Concomitantly, Neil Gershenfeld said in his book titled “When Things Start to Think” [46] that in the near future the things would be massively connected to the Net. Despite that, early projects and the embryonic concept of what IoT is nowadays come from 80’s, where the first connected devices (e.g., the Coke machine [124]) and papers forecasting ubiquitous computing through the connected and pervasive devices appear [129]. Indeed, the Genesis of IoT comes from the growth and maturity of multiple technologies such as Wireless Sensor Networkss (WSNs), embedded systems, RFID, data analytics, machine learning, among others. Such disciplines bring up advances in industrial and home automation, techniques explore devices limitation (e.g., size, memory, energy), scalability and robustness [28, 62].

The IoT has become popular since 2013 when several new sensor-based applications for the smart environment has gained attention [76]. Then, more attention has been given to connect heterogeneous devices to the global network which means a huge amount of devices being manipulated as well as being users of services on the Internet. Also, integrating IoT and mobile communication through cellular networks (e.g., 5G), cloud services (e.g., data storage, e-mail, text messages), and combinational cloud services (e.g., If This, Then That (IFTTT)¹ and Zapier²) will help to the flowering of IoT.

¹<https://ifttt.com/>

²<https://zapier.com/>

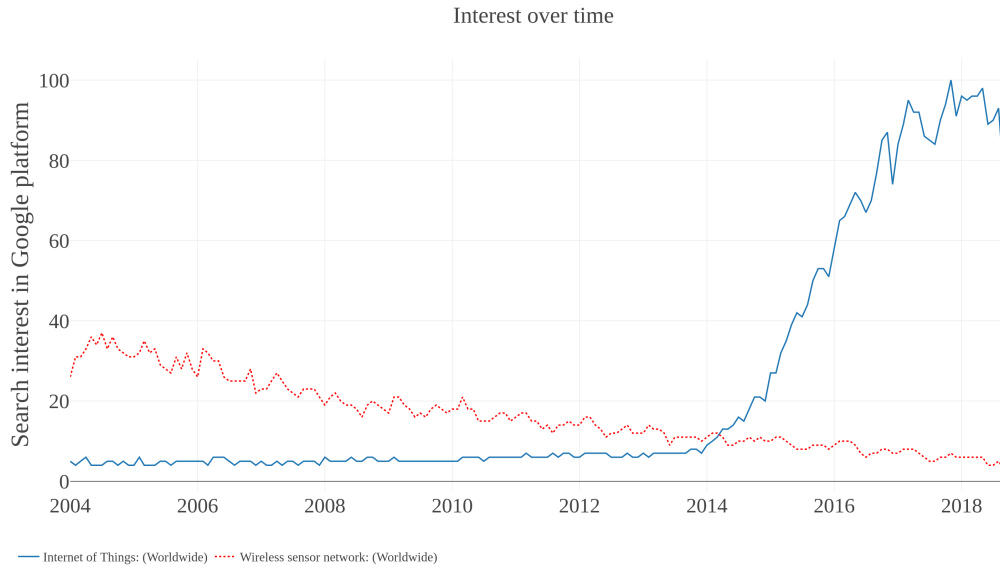


Figure 2.1: Interest for IoT and WSN terms on Google platform.

The IoT, in parts, comes up from the WSN, when these type of network have reached its plateau around 2010 [79, 111]. WSN have brought considerable advances in residential and industrial automation [28, 62], as well as techniques to efficiently explore devices constraints (e.g., energy and memory), scalability and robustness [79] in the network.

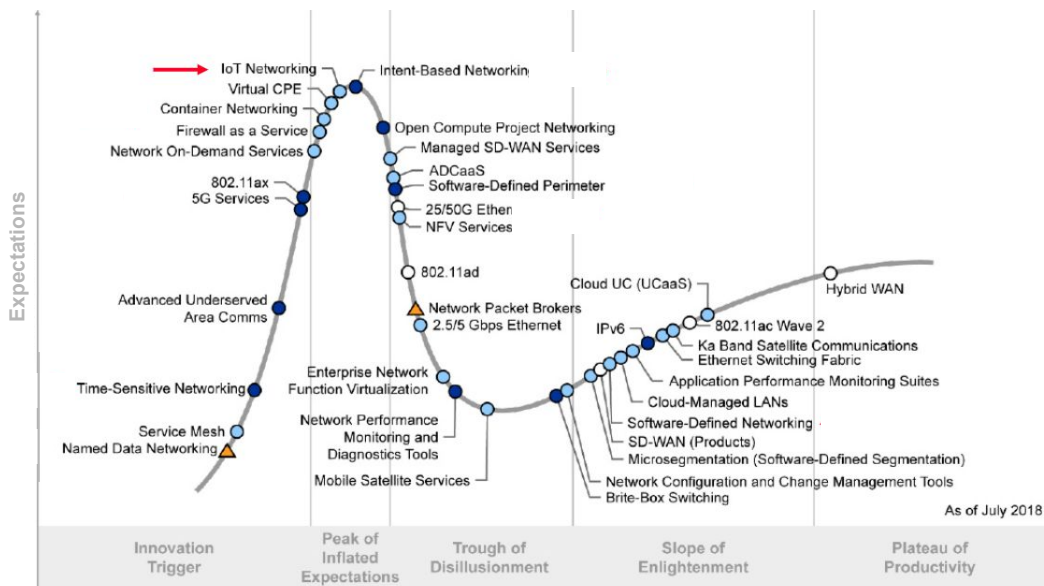


Figure 2.2: Hypecycle of July 2018. The IoT appears in the *Peak of Inflated Expectations* and IoT's Platforms appears as a *Technology Trigger*. Source: [45].

Specialists identified the term Internet of Things as an emerging technology between 2009 and 2012 [7, 45]. Figure 2.2 shows a diagram named *hype cycle* that displays the emerging, adoption, maturity and impact of a range of technologies. In July 2016, the term Internet of Things appears in the hype of inflated expectations meaning successfully stores of the technology used. Also, a volume of researches and applications were done to the IoT and then IoT's platforms appear as a result of the potential technology breakthrough kicks things off.

2.2 What is and Why The Internet of Things?

Internet of Things (IoT) is a network that extends the current Internet [117]. Nowadays, the core of the Internet includes millions of backbones routers and servers of high capacity. The Border (or fringe Internet) encompasses the major part of today's devices such as personal computers, and local networks linked to the Internet. It is estimated over 1 billion of such devices in the fringe Internet [117]. The IoT encompasses physical devices, wearable devices, vehicles, home objects and appliance and others embedded artifacts. In 2020, it is expected >30 billion devices connected to the Internet [90], and in long-term trillions of devices. It makes the IoT a challenge to the Internet perspective as well as a field of opportunities.

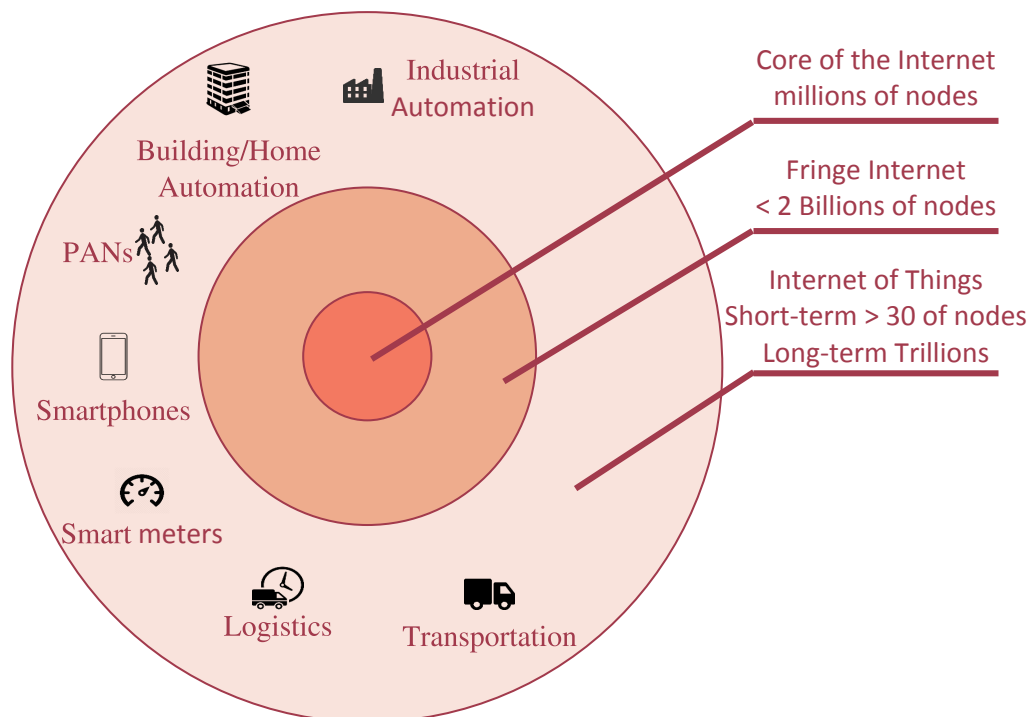


Figure 2.3: IoT as an Internet extension. Source: [117].

Naturally, interconnect billions, or eventually, trillions, of everyday devices to the global network is a challenging task. Especially, regarding that, the devices are heterogeneous in many dimensions (e.g., constraints, mobility, form, size). New IoT applications and paradigms also raise challenges and opportunities. For instance, new IoT applications may present different requirements to provide human-value, these requirements (e.g., specific data-traffic pattern) should be supported by the IoT infrastructure. New IoT paradigms such as the Internet of Mobile Things and Social Internet of Things are mobility-based. These new paradigms require support to mobile communication, mobile software, and hardware (see Section 1.1).

2.3 IoT's Basic Building Blocks

One can see the IoT as a set of complementary technologies aiming to integrate physic environmental objects to the virtual world. Figure 2.4 shows in rounded rectangles the IoT's basic building blocks:

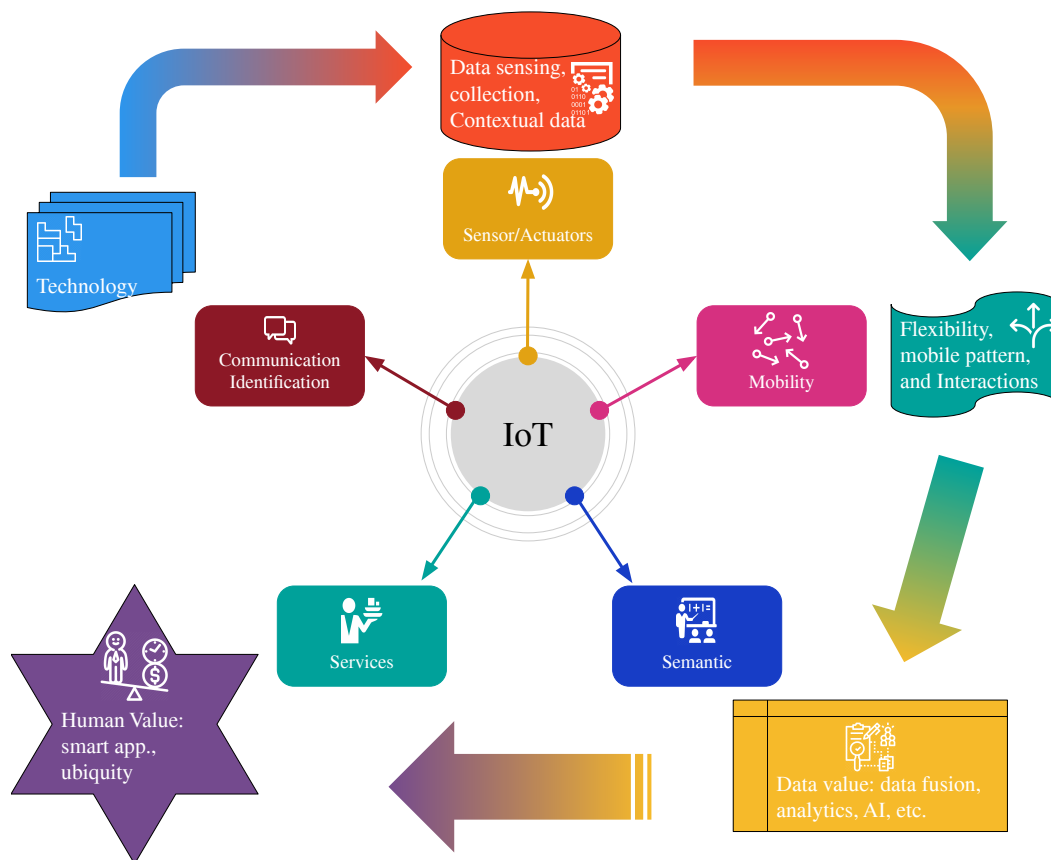


Figure 2.4: IoT's basic building blocks: from technology to human value.

- **Communication and identification:** as mentioned earlier, smart devices hold unit(s) of processing/memory, and communication giving them computational power. For processing, it typically uses micro-controllers, FPGAs, and tiny processors with a low power. For communication, often wireless technologies such as Wi-Fi, Bluetooth, IEEE 802.15.4 (low power links), RFID are employed. A critical factor is the identification of the objects which will be connected to the Internet and will form a network of >30 billion or even trillions of devices. So, it is a challenge to manage such a large network.
- **Sensors and Actuators:** it allows ordinary objects to sense its context. Its measurements may be used to infer changes in the environment. Thus together with computational resources, the objects can now perform smart tasks that were impossible before.
- **Mobility:** eventually smart objects will move freely, but nowadays they usually are carried on other mobile entities (e.g., humans and cars). The devices mobility patterns play a key role in the understanding how devices connect each, how long they remain connected (inter-contact time), and in the design of mobility management solutions. By handling mobility, the IoT takes a step towards ubiquitous computing, where everything is connected with everything at anytime and anywhere.
- **Semantic:** it refers to the ability to extract knowledge from smart objects. It includes resource usage and discovery as well as modeling information. It prepares the sensor data to be used in the services application by doing standardization through several web technologies such as Resource Description Framework (RDF), Web Ontology Language (WOL), and Efficient XML Interchange (EXI). The semantic is one of the key components of the IoT because it sends the demands to the right resource.
- **Services:** the IoT can provide four classes of services [5]: *identity-related services* those related to the association between real and virtual world; *information aggregation* aiming to collect and summarize sensor data; *collaborative-aware services* regard the decision made upon data analysis; and *ubiquitous services* providing services anytime and anywhere. Applications built on top of those services can vary widely (e.g., from identity-related services, information aggregation services, to collaborative and ubiquitous services.), and this makes the IoT a promising technology.

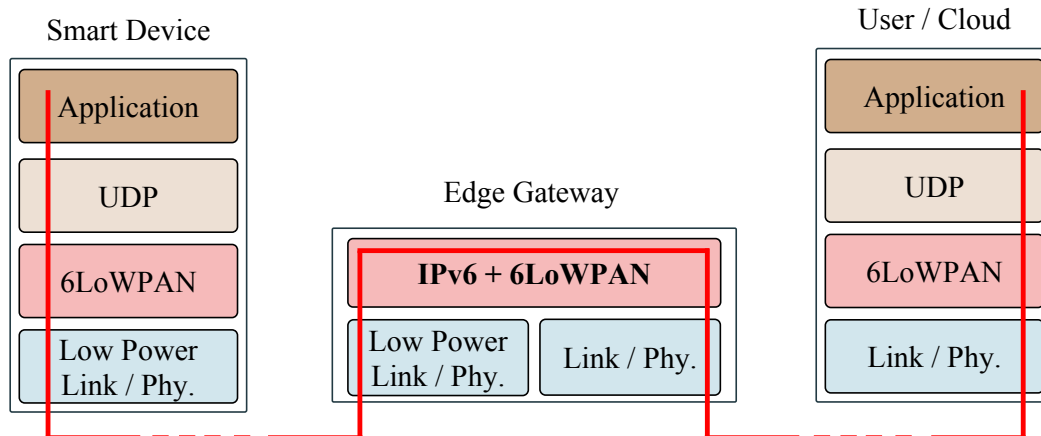


Figure 2.5: Smart devices interconnection with Internet example.

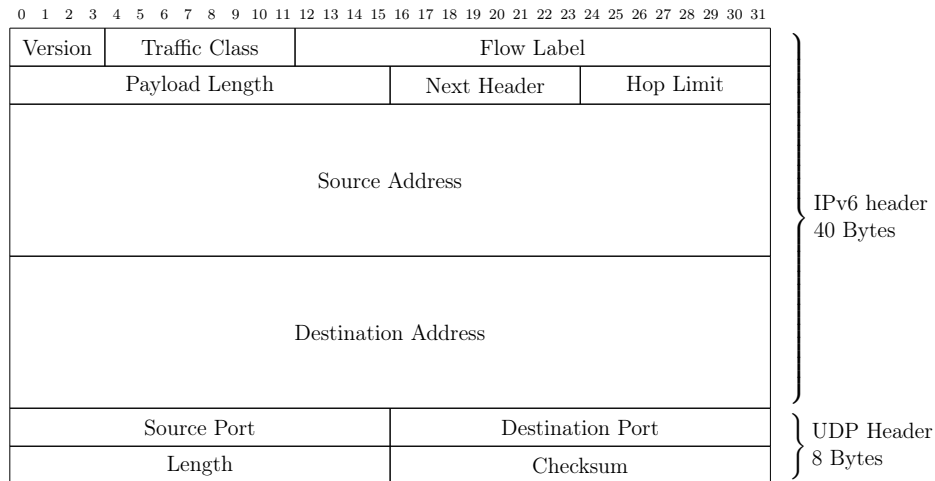
- **Power supply:** IoT's devices require a power source to operate. Although some IoT devices are directly connected to a wall outlet, others devices demand use of a battery (often non-rechargeable). Thus, selecting batteries for IoT devices can be tricky, as they serve to a range of application and there are a wide variety of devices with different shapes and sizes.

2.4 Internet Integration

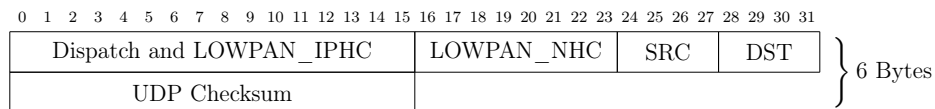
IoT is often considered a complex system. It is composed of many parts or elements that interact with each other. In order to integrate smart objects to the Internet world, it is often necessary for some kind of adaptations. Figure 2.5 shows the simplest view of an IoT architecture with some integrating Internet adaptation. It consists of three main tiers: smart devices (we have already highlighted in previous sections), edge gateway, and user or cloud.

IoT Internet integration was born from “the idea that the Internet Protocol (IP) could and should be applied to even the smallest of devices” [85], instead of make use of complex gateways necessary to translate between proprietary protocols and standard Internet Protocols. Though initially, IP was not the first thing that one thinks about sensor networks, the use of IP now is de facto the standard for IoT addressing. Firstly, the Internet Protocol version 6 (IPv6) is able to support the IoT's demand for scalability at the network level [117]. Another important benefit is that developers and users can use tools already developed for managing and debugging IP networks, cutting off the learning curve or developing overhead of an entirely new set of specific tools.

With this concept, 6LoWPAN was raised as a protocol definition for sending



(a) IPv6/UDP.



(b) IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN)/UDP.

Figure 2.6: Example of the IPv6 packet header with and without compression.

small packets IPv6 over LoWPANs [68]. The 6LoWPAN' basic goal was deal with some IPv6's requisites that did not match with low power links, for instance, Maximum Transmission Unit (MTU) and header sizes [32]. Thus, deal fragmentation and reassembly were one of the primary efforts because low power links typically have data units as small as 81 bytes[83], while a minimum IPv6 packet is 1280+ octets long. Also, as a frame of a low power network is typically 127 octets, and just the IPv6 header is 40 bytes, then only the remaining 41 bytes would be available for upper-layers protocols (e.g., UDP, TCP, HTTP or others), security and application data. Those and others issues (such as address autoconfiguration, security, network managements and others [83]) have pointed to the need for and adaptation layer and header compression.

The header compression relies on three major facts. First, the low 8 octets of an IPv6 can be the device's MAC address. Second, low power links frames (e.g., IEEE 802.15.4) carries this MAC address. Third, the IPv6's header fields are static. Shortly, by combining those characteristics, 6LoWPAN was able to compress 40 bytes from standard IPv6 packet header into a small version of just 2 bytes. Figure 2.6 gives an example of IPv6/UDP headers with and without 6LoWPAN compression technique. The LoWPAN header comprises a dispatch value identifying the type of header, then a byte indicating which of IPv6 files are compressed (LOWPAN_IPHC), followed by

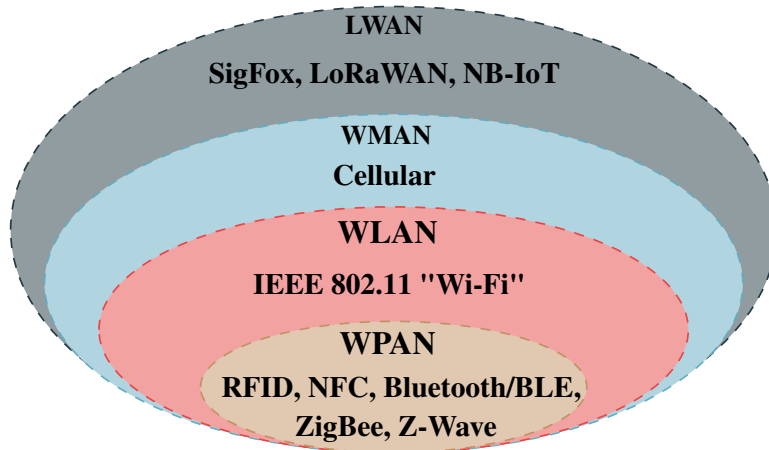


Figure 2.7: Classification of IoT wireless communication technologies by spatial coverage.

any IPv6 [83, 122]. In Figure 2.6b also shows the LOWPAN_NHC byte indicating a UDP as next header compressed, followed by UDP ports and checksum.

6LoWPAN was designed with special attention on IEEE 802.15.4 low power wireless links. However, it does not solve all problems and issues to run IPv6 over low power wireless links. Thus, there are several efforts to use IPv6 over different low power wireless links such as Near-Field Communication (NFC) [22] and Bluetooth Low Energy (BLE) [89].

2.5 Communication Technologies

In this section, we highlight the main interconnection technologies employed in IoT³. Wireless communication technologies vary in several dimensions such as communication range, throughput, energy efficiency, power efficiency and even in the typical topology [3].

Figure 2.7 shows an example of wireless communication technologies classification by coverage area. We deliberately present these technologies by dividing them into two main groups: long-range and short-range communication technologies.

Long-range technologies: this group is composed of Wireless Metropolitan Area Networks (WMANs) and Low Power Wide Area Networks (LPWANs) technologies that allow communication even in several miles away. Here, we highlight four trending long-range technologies: Cellular, SigFox, LoRaWAN, and Narrowband IoT.

³Note that the following list is a non-exhaustive one, more interconnection technologies and additional reading are available on [3]

- **3G/4G and 5G:** mobile network or just cellular network have boosted IoT adoption. They are already operational and provide communication from several miles away. The 2nd Generation of mobile telecommunications technology or just 2G was designed to voice, 3G for data exchange, and 4G for widely Internet access. Unfortunately, 3G/4G is not power efficient and usually tends to drain quickly energy resources from devices. Thus, often 3G/4G is not embedded into extremely energy constrained devices. However, 5G is about to arrive, it aims to address the limitations of the previous generation such as energy consumption, machine-to-machine communication, and speed (over $10\times$ faster than 4G). Also, 5G has been designed for mobile low-power and constrained devices as well. Thus, these 5G features have the potential key to enable IoT. IoT powered by 5G will use millimeter waves (for high speed communication and accommodate billions even trillions of devices), small cells (for connectivity), Multiple Input Multiple Output (MIMO), beamforming (for reduction of interference and improvements on efficiency), and full-duplex (more efficiency on communication).
- **Sigfox:** is a proprietary LPWAN technology that intends to be the first global IoT network focused on gathering data from billions of devices. Sigfox was designed to deal with a small amount of data. The technology acts as an Internet service operator for IoT and uses ultra narrow-band signals that pass through solid objects and propagates for long distances. Sigfox provides long range coverage from 30 km up to 50 km in rural zones, and in urban zones up to 10 km. The throughput varies from 10 bps to 1000 bps. Due to its specifications, Sigfox is suitable for use cases where data are gathered from devices such as remote water meters, temperature sensors. In other words, where users do not need to downlink to the devices. The proprietary aspect concerns the physical and MAC layers.
- **LoRaWAN:** is another proprietary LPWAN technology that intends to connect battery-powered devices to the Internet in regional, national, and global network. Unlike Sigfox, LoRaWAN provides by default bidirectional communication giving the opportunity of the device send data and be controlled, if needed. LoRaWAN data rates range from 0.3 to 50 kbps. The network operates in a star-of-stars topology, where end-devices are connected with one or many gateways which relay messages between end-devices and a central network server.
- **Narrowband IoT (NB-IoT):** is a Third Generation Partnership Project (3GPP) initiative for LPWAN, along with EC-GSM-IoT and eMTC which form

Cellular-IOT. It provides low cost and long battery life for IoT devices. Also, NB-IoT can co-exist with the previous generation of the mobile network by using principles from LTE physical layer to enable fast standardization and development. Thus, the coverage would be wide working both indoor and outdoor environments. Also, it is highlighted that over 10 years of battery life can be achieved in a wide range of use cases. NB-IoT provides downlink data rate up to 60 kbps and uplinks up to 50 kbps working in a star topology fashion.

Short-range technologies: Wireless Local Area Networks (WLANs) and Wireless Personal Area Networks (WPANs) technologies compose this group. We highlight the following trending technologies: RFID, NFC, ZigBee, Bluetooth, Zwave, and Wi-Fi.

- **Radio-Frequency IDentification (RFID):** the technology was designed for tracking and identification of stationary and mobile objects. Mainly, RFID is used on short-range communication where digital information is exchanged typically between mobile objects and stationary entities [69, 114]. Usually, mobile objects carry an inexpensive RFID tag where the digital data is encoded. Eventually, the data is captured by a device reader (RFID reader) through radio waves, such a device is attached to a more complex system. Those tags can be passive or active meaning that the former class can collect energy from RFID readers nearby and then operate by itself, while the latter (active tags) need a dedicated power source but can operate over wider distances from the RFID reader. RFID' tags operate into different frequencies that strongly determines their application options because of their connection range. Specifically, passive RFID' tags operate at three frequency ranges [128]: i) Low Frequency (LF) operates on 125-134 kHz reaching up to 10 cm; ii) High Frequency (HF) operates on 13.56 MHz⁴, it is capable of establishing communication between devices up to 30 cm far; and iii) RFID tags enabled to Ultra High Frequency (UHF) can be read from distances up to 100 m, UHF operates on 865-960 MHz.
- **Near-Field Communication (NFC):** the communication technology allows short-range communication over 13.56 MHz waves. NFC devices must be into few centimeters (<30 cm) way to communicate. Popular NFC uses are information sharing and contactless payment, where NFC enabled devices (smartphones, credit cards, key fobs) are put together boosting and automating their functionalities. NFC devices can work in three different modes: i) *Card emulation* where

⁴The same frequency of as NFC technology.

devices (e.g., smartphones) play the role as smart cards being able to accomplish transactions (e.g., payments); ii) *Tag reader or writer* is the mode that enables NFC devices to read information encoded on tags attached on other devices; And iii) *Peer-to-peer* that allows devices to exchange information and share files in an ad hoc fashion.

- **BLE & Bluetooth (BT):** such wireless technology has gained a ton of popularity with smartphones and several wireless gadgets. The BT protocol has been designed by Bluetooth Special Interest Group (Bluetooth SIG) and has become a key technology for WPAN. Shortly, BT is divided into two groups: Classic and Bluetooth Low Energy (BLE). The classic group is composed of 3.0 or early versions that were designed to establish a connection and keep a stream of data between the devices. The latter, BLE group is composed of versions 4.0 or later that have been optimized to low power consumption. Classic and BLE are not compatible, thus some devices may have both. Classic BT can deliver data rates up to 2 Mbps, while BLE up to 1 Mbps. Communication range is the same for both, technically, they can communicate up to ≈ 100 m of distance but, in reality, 10-20 m are reachable. BTs groups also have similarities such as both use 2.4 GHz, it is employed for short-range communication and typically use the star topology. However, BLE introduces low power, more sleep modes, faster pair/connection times, allows mesh topology (versions 4.1 or later), can connect to the Internet but requires a gateway.
- **ZigBee:** is a suite of protocols designed to low power wireless devices. It has been developed by Zigbee Alliance and it is based on IEEE 802.15.4 specification. Thus, it aims to provide low-cost and low power networking. The technology can operate in three different topologies star, mesh, and tree. Depending on power output transmission, ZigBee can establish communication typically up to 100 m indoor and 300+ m line-of-sight environments, operating on 2.4 GHz, also ZigBee provides a data rate up to 250 kbits/sec.
- **Z-Wave:** is a wireless communication protocol developed by Zensys. Their main idea is to propose an alternative solution flexible enough to Z-Wave devices be reachable from or access the Internet, operate in a mesh fashion instead of the star topology and allow wider communication range (up to 100 m) than BT technologies. Z-Wave operates on the 800-900 MHz radio frequency, it can deliver up to 40 kbits of data rate.

- **Wi-Fi:** this wireless communication technology is the most popular due to their presence in everyday environments such as our homes, offices, industries, and others. Wi-Fi follows the standard IEEE 802.11 for transmission and codification. The standards have been developed since 1997 providing several versions and typically operating on 2.4 GHz or 5 GHz waves. Depending on the Wi-Fi version, it is possible to deliver data rates raging from 1 Mbps up to 20 Gbps over respectively the versions IEEE 802.11-1997 and the predicted IEEE 802.11ay. Although IoT devices have been employed with Wi-Fi capabilities, the technology is not great for most IoT constrained battery-powered devices that need to operate for months or even years. However, some Wi-Fi versions fit IoT's demands. They are named Wi-Fi HaLow (802.11ah) and White-Fi (802.11af). Wi-Fi provides connectivity up to 20-250 m depending if the environment is indoor or outdoor and the power output, we highlight that Wi-Fi HaLow can reach up to 1 km with the right antenna.

2.6 Mobility Meets The Internet of Things

In a mobile-enabled IoT, it is important to understand what the devices do, where/when/how they do, where they move for, what are the places where they come and go across the time (hourly, daily, etc.). In that context, users should have a consistent experience of use even in occasionally connected set devices. Recent research questions have focused on the mobile aspect of the IoT's devices [87]. Examples of devices are robots, Unmanned Aerial Vehicles (UAV), people carrying smart devices, and vehicles. These new mobile technological artifacts rise several challenges not faced by the traditional static IoT such as mobility induced sensor network design, robustness over mobility, coordination, concurrency, opportunistic computing, and others [86, 87, 120].

Mobility concerns three significant aspects (see Figure 1.1): The mobile hardware that includes the devices and components that provide or access the IoT services. Typically, these devices have a wireless transceiver enabling them to communicate with other devices and, eventually, they can move by itself or being attached to a mobile entity. The mobile software encompasses the operational software framework (OS, applications, etc.) able to providing support to mobility applications. The mobile software runs on top of mobile hardware, therefore, it should be aware of the hardware constraints while dealing with the requirements and characteristics of the IoT applications. The mobile communication refers to data exchanges, seamless and reliable communication. Also, mobile communication should provide support to transparent

communication although occasionally the devices may be detached from the network.

2.6.1 New IoT Paradigms and Requirements

A natural evolution of static IoT is the smart object mobility freedom allowing new IoT paradigms such as the Internet of Mobile Things (IoMT) and Social Internet of Things (SIoT). For IoMT, the major changes appear from the highly dynamic network formed by mobility-enabled devices. While SIoT pushes forward the concept of everything real becomes virtual, meaning that each person or object has a virtual entity counterpart on the Internet. Then, as in the real world, these virtual entities can consume and produce services as well as do social ties among them.

The IoMT paradigm encompasses the smart “things” able to move independently or be attached to a mobile entity, and these “things” remain accessible remotely from the Internet whenever and anywhere if one wishes. IoMT faces challenges different from static IoT, for example, *context and mobility pattern recognition*, *connectivity under high dynamic links*, *energy availability*, and *security and privacy*. Few recent studies have focused on the IoMT, especially regarding the framework support to this new paradigm [107, 120]. Indeed, most of the work present predictions or assume that a basic support framework exists, for example, routing under the mobile scenario is well understood. However, as mentioned, several challenges still need to be solved.

SIoT extends the original IoT by enabling smart devices to establish social ties. Note it does not mean that smart objects will sign a Social Media, but they will be able to create social relationships among them. As pointed by [9], SIoT rise several advantages such as easy objects and services discover without human involvement by just using the object’s relationships (e.g. gadgets of Bob can configure new Bob’s gadgets). Another benefit is that different levels of trustworthiness and interaction can be applied depending on the friendship between objects, enabling humans, for instance, to measure security levels of the relationship of the objects. [9, 10] list five types of smart objects relationship of a SIoT:

- Parental Object Relationship (POR): relation established among objects from the same production batch, e. i., objects from the same brand and produced in the same period;
- Co-location Object Relationship (C-LOR): this relation occurs among objects always used in the same place. For instance, objects from a home environment. They can be heterogeneous and from different brands;

- Co-work Object Relationship (C-WOR): occurs when objects collaborate to provide a common application;
- Ownership Object Relationship (OOR): relations between objects belonging to the same owner, e. i., smart-phones, laptops, video game consoles, and others.
- Social Object Relationship (SOR): the social relation may appear when objects contact other objects sporadically or continuously. For instance, devices from classmates or people that work in the same place.

To support IoMT and SIoT applications and turn them a reality several challenges need to be overcome. In the following we list some of these challenges:

- Communication requirements: many mobile-enabled IoT applications will require any-to-any communication due to the flexibility offered. For example, devices in a SIoT will need communication in a bi-directional way to create their social tiers as well as provide and access services. The challenge is the high dynamic and the device heterogeneity turning the any-to-any communication a hard task. In Chapter 6 we present an alternative to provide any-to-any communication under static and link fault networks as well as in mobile highly dynamic networks;
- Dynamic network topology: mobile entities in the network cause high dynamic topologies. Make this dynamic aspect transparent to the applications is a challenge. For example, mobility can cause disconnections, routing address identification changes. In Sections 5.3.4 and 6.3.3.1, we propose alternatives to overcome respectively temporary link dynamic failures, detect motion, and reconfigure routes to reflect entities in its new position in the network;
- Opportunistic computing: this field is concerned with how devices can interact with each other opportunistically. With IoMT, people can interact with a stationary or mobile sensor rising the Opportunistic IoT (OIoT) [50]. There are several traditional opportunistic network solutions for routing, data dissemination, incentive mechanisms, security, trust, and privacy. However, such solutions will need to be revised or new solutions proposed to overcome the constraints posed by IoT contexts such as energy and resource limitations.
- Mobility patterns: unfortunately, data to study and better understand device mobility patterns in the context of IoMT or even SIoT are not available to date. When there is, they have limited scope or are not available for free study [66].

Thus, authors of proposed solutions have made several assumptions in their design of mobility-aware solutions in IoT. The mobility pattern can vary widely, for example, vehicles travel in specific pre-defined patterns while pedestrian carrying smart devices. Modeling these patterns and understanding their variability would yield valuable insights to propose efficient solutions: efficient energy consumption, network connectivity, and coverage, opportunistic contacts, and others. In Chapter 6, we introduce an alternative routing protocol that considers human and non-human mobility through mobility models in the experiments, which shows an option to design mobile-aware routing protocols.

Note that other challenges arise when IoT's devices can move. For example, device indoor localization, data gathering, and in-network data fusion. [9, 87, 120].

2.7 Concluding Remarks

Throughout this chapter, we have given an overview of the historical perspective, the fundamental concepts and building blocks of IoT, as well as IoT IPv6 integration and different communication technologies. Also, we presented the mobile paradigm within IoT's context, which brings new IoT paradigms (IoMT and SIoT) as well as a range of research challenges and open issues.

Chapter 3

Model and Definition of Problems

In this Chapter, we declare our problem set to which we have proposed solutions as well as the model to evaluate our propositions.

3.1 Definition of Problems

As our focus is in the IoT routing layer, let us briefly revise what this layer does. Despite, we are hiding low-level details, addressing and routing are two of the most important task of the routing layer [98, 121]. The former deals with identifying devices into the network typically using the IP. However, in IoT networks such a protocol suffered some adaptations forming the 6LoWPAN. The latter (routing) concerns to getting messages/packets from the source all the way to the destination, typically requiring many hops during the routing process. We focused on issues concerning the routing part.

In IoT, the routing task becomes an issue because devices are typically constrained in several ways and may be mobile. Moreover, they have scarce energy and computational (CPU and memory) resources. Also, the devices can move around causing topology dynamics. Thus, those factors call for solutions specially designed for them. The main issue in the routing task is to provide a routing mechanism by making use of the lowest amount of resource even in mobility events.

There are two main classes of routing protocols: pro-active and reactive¹ [104]. The former attempts to maintain consistent, up-to-date routing information to provide at least one communication paradigm. The latter creates routes only when desired by the source node. In this work, we focus on pro-active protocols.

¹One can also find pro-active as table-driven and reactive as on-demand protocols.

Three main communication paradigms are mainly employed by routing packets in an IoT network: many-to-one; one-to-many, any-to-any. The paradigms differ significantly, affecting their implementation code complexity (e.g., memory, energy), application design, and applicability. In the literature, there are IoT routing protocols based on at least one of the communication paradigms. Below, we define each mobile communication paradigm and we point out well-known protocols that are a reference in each paradigm:

- **Bottom-up (or Many-to-one):** this communication paradigm (often called as data collection paradigm) is the most known and well-solved. It supports applications that need to gather data, at a single point, from the networked devices. The main issue is to create and maintain (pro-actively or on-demand) the routes from the devices to a single target (usually, the root or base station).

Two examples of protocols for many-to-one communication paradigm are: Collection Tree Protocol (CTP) [47] and Routing Protocol for Low-Power and Lossy Networks (RPL) [133]. Chronologically, RPL was standardized after the CTP proposal. Essentially RPL implements CTP’s functionalities, but it also provides other features, i.e., multiple routing trees (named Destination Oriented Directed Acyclic Graphs (DODAGs)) [133].

- **Top-down (or One-to-many):** well-known as data dissemination paradigm, it has characteristics that are opposite to the previous one. It supports applications that need spread (disseminate) some information from one device to other networked devices. In general, they disseminate data/code to set/reset the device reconfiguration. The main problem is the efficient way to propagate the information to all network devices being aware of duplicate re-transmissions, receptions, and reachability.

Deluge [21] and Trickle [75] are examples of data dissemination protocols. The first one was built to disseminate a large amount of data from one or more source devices to all other networked devices. The second one uses a “polite gossip” policy that broadcasts information (typically of small size) to local neighbors of the source device, but it remains silent if it receives identical information from its neighbors.

- **Any-to-any:** this paradigm supports both paradigms and allows communication between any two devices in the network. It does not present significant restrictions for traffic exchange unless the device has limited computational resources. In particular, memory might be a problem since it is necessary to maintain routes

for all devices reachable in the network. An example of an any-to-any routing protocol is the RPL which is considered the standard routing protocol for IoT.

We are especially interested in this mobile communication paradigm because of its applicability and potential to enable new mobile IoT applications. In Chapter 6, we propose a routing protocol able to perform any-to-any routing under mobile and static scenarios.

- **Intermediate approach:** Often, an intermediate approach can be useful where bi-directional communication between the border router and devices are required, and when devices are too constrained to accommodate any-to-any implementations. For instance, applications that require an acknowledgment for delivered messages. In this direction, it is possible to combine the paradigms one-to-many and many-to-one and create an intermediate one. The protocol eXtend Collection Tree Protocol (XCTP) [113] is an example of this approach. Indeed, XCTP is a CTP's extension that keeps all CTP features, but provides reverse routes (from the border router to the devices).

In mobile scenarios, besides maintaining consistent and up-to-date routing information, it is expected that the routing mechanism deals with frequent topology changes. Normally, IoT routing protocols rely on timers² that trigger beacons³ to check topology changes. Those timers play an important role in energy resource usage. A basic trade-off is faced here if the timer triggers several beacons, then it will quickly detect topology changes, and energy resources will be quickly drained. Otherwise, if the timer sends a few beacons, then it will save energy but it will be unaware of topology changes. Thus, routing solutions for mobile scenarios should be aware of this timer trade-off.

In summary, two main problems are addressed in this thesis: (i) To pro-actively construct and maintain routing information even in case of mobile events in an IoT network; and (ii) To balance the trade-off faced by the beaconing timer.

3.2 Models

We have some assumptions and used models to study those problems. Despite keeping our propositions as simple and generic as possible, our results were evaluated mostly under a simulated environment. Below, we highlight our assumptions with respect to the hardware and models to build our solutions presented in the following chapters.

²Also known as Beaconing Timers.

³Small packets to check some properties. In this case, reachability.

3.2.1 Entities

“Things” or just entities⁴ are the generic names when referring to the devices employed in the IoT. Entities are of different types (e.g., TVs, home cleaners, vehicles, and smartphones). Also, entities have a context (e.g., a place, a state). Here, we are interested in the mobility context. Although mobility can be studied in a broad spectrum, we just kept the mobility context as simple as possible by classifying entities into two main groups: those having mobility or no. For mobile entities, we divide them into human-like and non-human mobility behavior. The context of the entities is obtained through sensing elements that also have a wide spectrum, e.g., from sensors like GPS or accelerometer to social media sensors [16, 118]. Aware of the entities’ context it is possible to design custom solutions as presented in the following chapters proposals.

Our entities assume a basic unit of hardware that holds at least the following characteristics: (i) Processing unit(s); (ii) Memory unit(s); (iii) Communication unit(s); (iv) Sensor(s) and/or actuator(s). (v) Energy source. One can find basics device architectures in the following literature [79, 105]. For entities with those characteristics, we also named them as smart objects.

The set of sensors and actuators may play a key role in the development of solutions. For example, with a dedicated proximity sensor, it is possible to detect entities’ proximity easily. We do not make any assumptions on the set of resources, except in Dribble (see Chapter4) re-evaluation phase where GPS reads was employed to make the solution simpler, although approaches with no GPS can be built.

3.2.2 Mobility and Dynamic Model

To the best of our knowledge, there is a lack of real and diverse mobility and topology dynamic traces for IoT’s entities, usually due to privacy-related or technical issues. To overcome such a situation, researchers have developed mobility models to fill this gap [53, 84]. Those models try to mimic real mobile entities behavior allowing us to generate variable traces in terms of space, time, and size.

We employ mobility models in our simulated environment classifying them into two groups: human and non-human patterns. For human mobility model, we highlight the Small World in Motion (SWIM) [81] and the Group Regularity Mobility Model (GRM) [91]. SWIM produces synthetic traces with similar properties of real mobility traces. It assumes that humans go to places near their home, where they meet oth-

⁴In some moments, we also named entities as *node*. This is a usual term employed in computer networking, especially concerning the routing layer.

ers, and, eventually, return to their homes. GRM presents similar properties, but it introduces the dynamics of group meetings and social community structure.

Over the years several non-human mobility models were proposed [6, 12, 84]. We highlight the Random Waypoint (RWP), a well-known mobility model to evaluate MANET routing protocols [12]. In RWP, the entities move freely in a random direction, velocity, and acceleration. Also, there is an RWP extension named Cyclical Random Waypoint (CRWP) [106], where the entities behave similarly as in RWP. However, for CRWP entities, after n chosen destinations, the mobile entity returns to its initial position. CRWP is useful to model scenarios where some entities move to different destinations, and eventually, return to their initial positions, which is the case of objects (e.g., portable devices and environment cleaners) that move in homes, offices, universities, hospitals, factories, among others.

Each mobility model has its parameters. In timely, we highlight each parameter (number of nodes, environment dimensions, statistical parameters, and others) putting them into the context of our evaluation environment setup.

To the best of our knowledge, there is no widely and well-known trace of link dynamics in IoT context. To evaluate our solutions, we simulated a range of faulty scenarios based on experimental data of TelosB sensor motes deployed in an outdoor environment [11]. Randomly devices have their radio turned on/off during a time interval uniformly distributed. It simulates network topology changes. In this thesis, we highlight the parameters used in our experiments.

We make use of those models to evaluate our proposed solutions for mobile and dynamic environments. In general, we use two groups (human and non-human) and faulty scenarios in our evaluations, so we use mobility and faulty contexts that are simple and treatable. Also, we highlight that mobility patterns and faulty studies in IoT should be further investigated, but they are out of scope of this thesis.

Chapter 4

Mobility Detection

4.1 Contextualization

Smart devices have been part of our daily routine. They can be attached to infrastructures, wearable, and be moving by itself. When those devices are networked and connected to the Internet, they form the so-called IoT. Nonetheless, they introduce new challenges from the network lens, because they are heterogeneous (e.g., TVs, smartphones, vehicles, etc.) and have different degrees of mobility. Thus, manageability and scalability are examples of key issues that ask for solutions, especially when the mobility factor is present. Recently, most of IoT's solutions were proposed for static networks [57]. Only a few attempts took the mobile context into consideration [92, 106]. In the mobile and wireless environment, the routing protocol is a key component to enable mobility to the IoT. Mostly of routing protocols for mobile IoT have one timer scheme that governs the communication structure construction and maintenance by triggering from time to time control advertisements.

The timer scheme must deal fairly with a basic trade-off. If it is too greedy by sending advertisement frequently, it responds quickly to topology changes, but it spends energy and introduces an overhead to the wireless shared channel. However, if the timer scheme is too slow, i.e., it sends advertisement infrequently, it will save energy and bandwidth, but topology problems will persist for a long time.

To balance this trade-off, we propose Dribble, a learn-based timer scheme selector. To the best of our knowledge, IoT networks are governed by a single timer scheme, without concerning devices mobility. Dribble differs from single timer schemes by setting a custom-made timer scheme to devices conveniently. It does that through a learning process of devices' mobility patterns and matching them to a proper timer scheme. We evaluate Dribble against traditional timers such as Trickle Timer (TT)

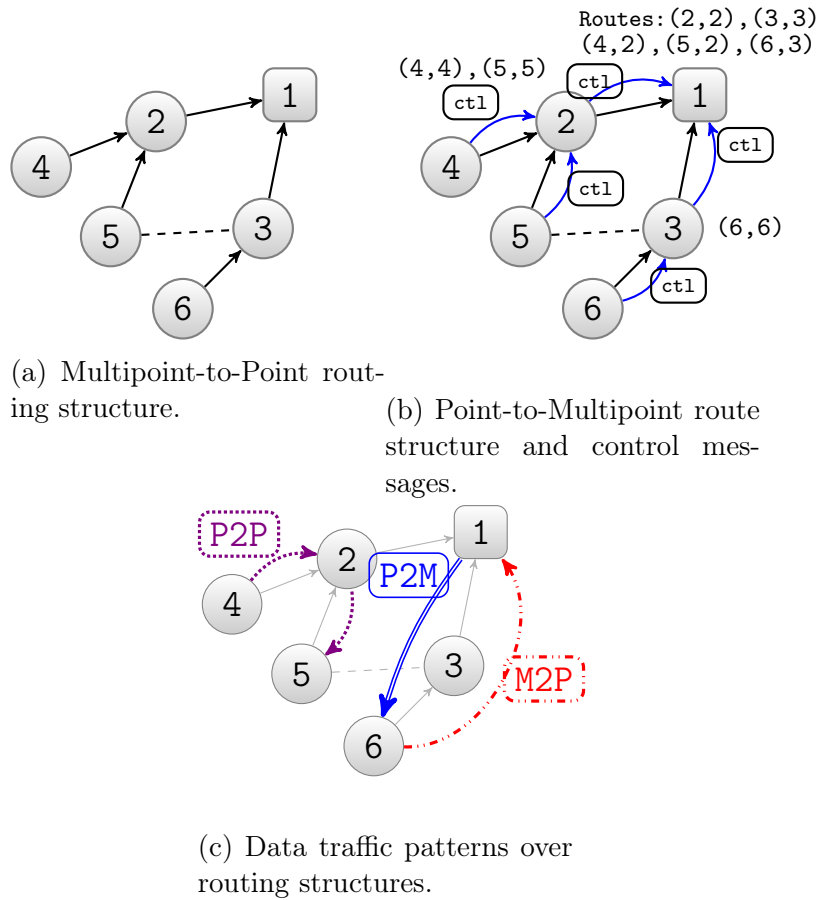


Figure 4.1: (a) the Multipoint-to-Point (M2P) routing structure that creates routes from nodes towards the root. (b) presents the Point-to-Multipoint (P2M) routing structure typically created through control packets sent over (a) structure. (c) the three main data traffic patterns over the routing structure constructed on (a) and (b).

and Reverse Trickle Timer (RevTT) through extensive simulations. Our analyzes show that Dribble presents a better trade-off balance than a single timer scheme.

The rest of the chapter is organized as follows. In Section 4.2 we review routing in IoT. Section 4.3 states the timer scheme problem and describes the related work. In Section 4.4, we present Dribble workflow. Dribble evaluation is detailed in Section 4.5. Finally, in Section 4.6 we highlight the final remarks and conclusions.

4.2 IoT Routing in a Nutshell

In IoT, mobility-enabled routing protocols build structures to perform multi-hop data forwarding in three main traffic patterns: Multipoint-to-Point (M2P), Point-to-Multipoint (P2M), and Point-to-Point (P2P). Figure 4.1a shows the routing structure required to provide M2P data-traffic. Typically, the routing protocol creates a rout-

ing Directed Acyclic Graph (DAG) oriented towards one or more special device node (known as a border router or base station). By default, each node maintains at least one preferred parent, which is used to forward data. The M2P traffic pattern is cost efficient in terms of memory requiring few routing states.

P2M is the dual traffic pattern (from root to nodes). Protocols that support P2M build paths by using extra control packets and routing states. Figure 4.1b shows how a protocol can create P2M paths¹, initially it is created a DAG, and then periodically each node reports to the direct parent, through control packets, reachable downwards nodes. By supporting both previous traffic patterns, a protocol can combine them to provide P2P communication (Figure 4.1c). Several routing protocols were proposed to support at least one IoT data traffic pattern (e.g., XCTP [113], CodeDrip [59], and Mobile Matrix [106]). RPL [133] became *de facto* the standard routing protocol for IoT. RPL is a distance vector protocol that builds a DODAG.

RPL introduces three main control packets to build and maintain routes [133]. Figure 4.2 show the RPL control packets exchanges flow. i) DODAG Information Solicitation (DIS) is used by a new node to ask for valid DODAG instances nearby in its neighborhood; ii) DODAG Information Object (DIO) carries information allowing new nodes to discover and join to RPL's DODAGs instances and get its configuration parameters. Initially, the root fires the first DIO triggering a DODAG formation. Also, a DIS packet reception by a node triggers a DIO transmission; iii) DODAG Destination Advertisement Object (DAO) propagates destination information upwards through node's parent along the DODAG. If RPL is on store-mode², child nodes send unicast DAO to selected parents. In Non-Store mode, DAO is unicasted to the border router; iv) DODAG Destination Advertisement Acknowledgement (DAO-Ack) is an optional control packet that is acknowledged back to the sender of the DAO.

Most of IoT routing protocols (e.g., RPL and Mobile Matrix) relies on two key components to construct and maintain their routing structures: a routing metric and a timer scheme. The former, it is focused on expressing the link quality which is typically a wireless link. Well known IoT routing metrics such as ETX and 4-Bit are useful to catch the highly dynamic and busy behavior of wireless technologies employed (e.g., IEEE 802.15.4) [47, 106, 133]. Such link metric is regularly advertised into control packets (e.g., DIOs) helping nodes to find best paths. The focus of this work is on the timer schemes which concern the triggering time interval between consecutive control

¹Note, there are others approaches to build P2M as presented in [92, 106, 133]

²RPL supports two types of downwards routing schemes named Store and Non-Store Modes. In Store Mode, relay nodes in-network can keep downwards routing structure, i.e., keep routing tables. In Non-Store Mode just the border router node keep downwards routing structure [133].

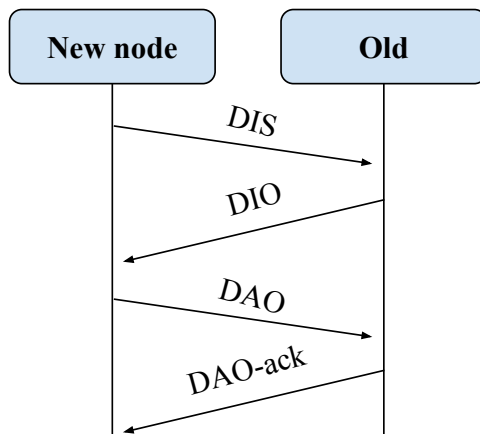


Figure 4.2: RPL control packets.

packets advertisements. The timer scheme plays a key role by tracking topology changes and inconsistencies (e.g., mobility and loops respectively) as well as energy consumption and channel occupation overhead. Following, we highlight the problem tackled by timer schemes and existing solutions.

4.3 Related Work and Problem Statement

Mobility is a major factor present in everyday life, thus the mobility of “things” is a natural event in the cyber-physical space. In IoT, even reduced device mobility can cause topology changes due to, for instance, the short-range wireless link technologies (e.g., IEEE 802.15.x). In such a situation, the routing protocol must rebuild routes to reflect the new topological organization as soon as possible mitigating network disconnections. Typically, pro-active routing protocols advertise control packets from time to time to build and manage their routing structures. In RPL, a node can re-attach to a DODAG upon receiving a DIO (upwards routes control packet) and then it transmits DAOs to build downwards routes.

The timer scheme governing the control advertisements plays a fundamental role by impacting in the network overall performance. First, the responsiveness to topology changes. Second, the energy consumption upon sending control packets. Third, the introduction of channel overhead also by firing control messages. In Figure 4.3, we put all together as a basic trade-off faced by time-schemes. If the timer scheme frequently schedules advertisements, i.e., it has a small interval between consecutive beacons, it can quickly respond to topology problems at a high cost of energy expenditure and

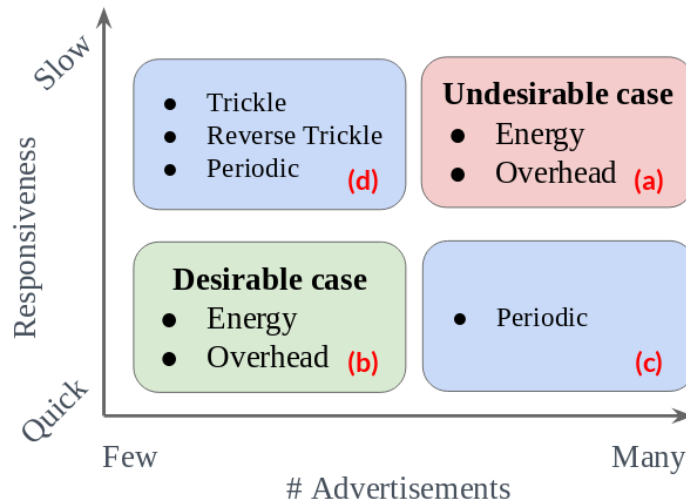


Figure 4.3: Timer schemes' basic trade-off faced.

channel overhead. On the other hand, if a timer scheme occasionally (large interval) advertises control packets, devices use less energy and introduce low overhead, but topological inconsistencies will persist for a long time. As shown in Figure 4.3a, it is undesirable to create a timer scheme that is slow to figure out links change and sends many advertisements. On the other hand, it is desirable a timer scheme that sends a few control packets advertisement while is fast to respond to topology changes (see Figure 4.3b). Thus, our effort is in propositioning a solution with such characteristics.

In the literature, few attempts have been made to fill this gap, especially in IoT environments where mobility is present. The vanilla approach periodically sends advertisements in a fixed interval see the pattern in Figure 4.4 (Periodic). If the advertisement rate is high then the routing protocol will be able to quickly track topology changes, but a high channel and energy usage are required. If the advertisements are sent with large interval low energy and channel occupancy are required, but the routing protocol will be slow to detect topology changes. Periodic timer scheme leaves to the network operator the responsibility to control the trade-off, thus the scheme appears in Figure 4.3c,d.

Often, routing protocols, as is the case of RPL, use the adaptive beaconing Trickle Timer (TT) algorithm [75]. TT advertises control packets faster when topological inconsistencies occur in the routing structure, otherwise, it decreases the advertisement rate exponentially as shown in Figure 4.4 (Trickle). In such a strategy, it is assumed that the network will be stable, i.e., it is almost static with just a few link changes. Otherwise, the timer scheme will remain in the minimum interval (typically a very small interval) implying in network overload of control packets and energy expenditure. The RPL specification suggests 8 ms and 2.3 h as a minimum and a maximum period

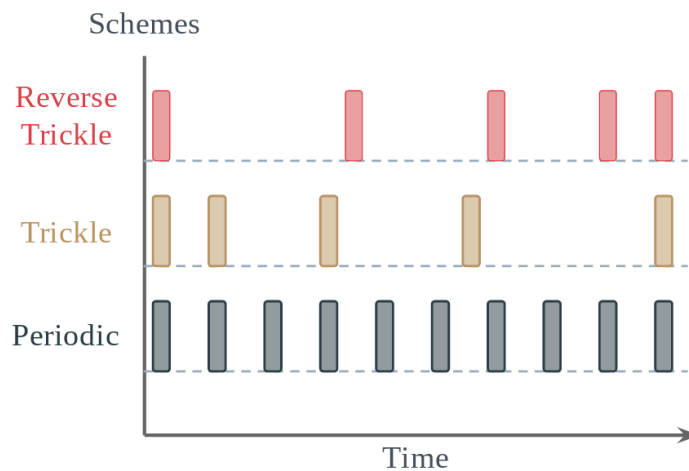


Figure 4.4: Three main timer schemes commonly used in IoT routing protocols.

respectively for TT [133]. Therefore, a node can wait for a relatively long time before sending a control packet, which turns the tracking of topology problems an issue to standard RPL in mobile environments, for this reason several RPL’s adaptations were proposed for mobile scenarios which typically changes TT scheme [92]. The TT is positioned in Figure 4.3 (iii) given its characteristics.

Reverse Trickle Timer algorithm [25] works in an opposite fashion than TT as shown in Figure 4.4 (Reverse Trickle) It starts with a large interval between advertisements, and then it halves the interval after each advertisement is fired, i.e., increasing the advertisement rate. The authors argue that RevTT is suitable for mobile networks where a mobile node attaches to a parent (preferably static). It is assumed that when a node connects to a parent, such node will likely remain connected to this parent for a long time. As long as the node remains connected to the same parent, it is likely to that node moves away. This justifies a large initial interval and the short interval after long periods. The authors evaluated RevTT with RPL in a mobile network environment.

Besides those approaches, there are some proposals in the literature that make hard assumptions (e.g., knowledge about parent position) to operate [44, 92]. Also, vehicular networks is another field which widely explore the benefits of adaptive beaconing approaches [52, 116, 119]. Those proposals focus on increase delivery rate and reduce the channel overload which is important for safety applications and routing protocols. However, they were not proposed to constrained IoT’ devices.

Different from previous proposals, we do not propose a new timer scheme. Instead, we propose a selector to set a custom timer scheme given the mobility pattern of an IoT device. We advocate that by matching timer schemes to well-known mobility patterns, it is possible to balance the posed trade-off.

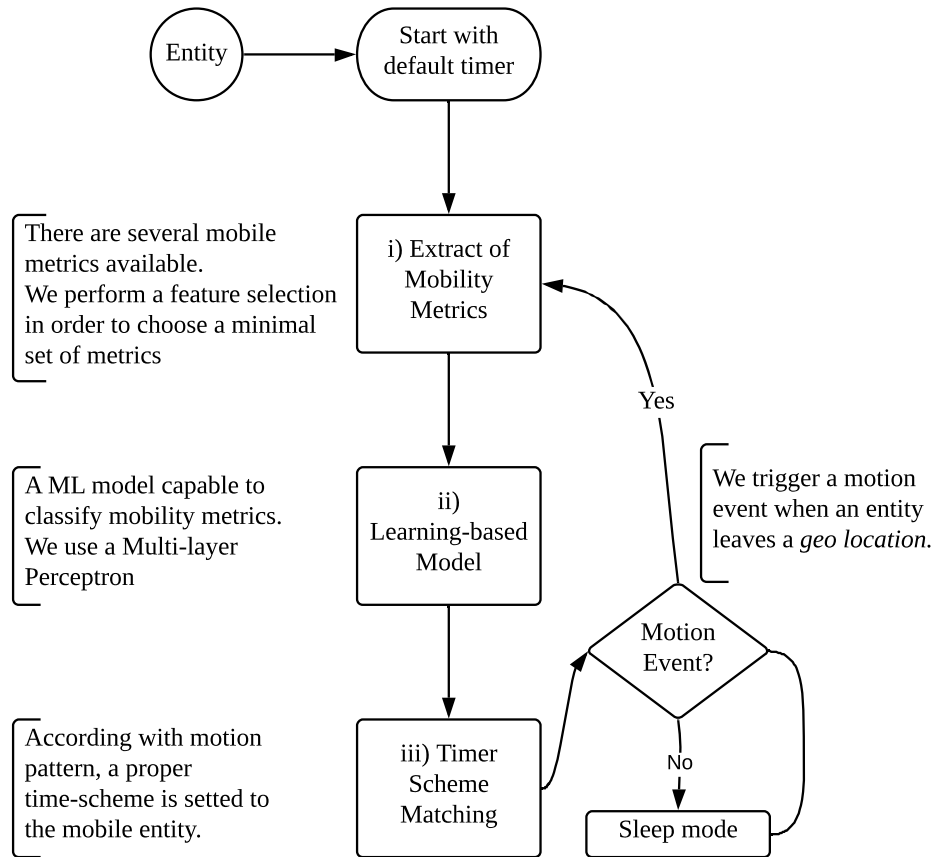


Figure 4.5: Dribble flowchart diagram.

4.4 Dribble Design

In this section, we detail Dribble, a learn-based timer scheme selector that helps IoT routing protocols to manage the mobility. Dribble learns the devices mobility patterns and then it assigns a custom timer scheme to each IoT device in a distributed fashion. We advocate that it is possible to better balance the basic trade-off faced by a single timer scheme (see Section 4.3 and Figure 4.3). We have designed Dribble to be as generic as possible by no assumption about fixed topologies, network range or sizes in its methods. The rest of this section is guided by the flowchart presented in Figure 4.5 that present Dribble's algorithm.

Figure 4.5 shows the Dribble high level work-flow process. It consists of the following main stages that each entity (device) must execute: A) *Update of Mobility Metrics* that are extracted from the entities allowing particular characterization; B) *A learn-based model* (e.g., a neural network) which aims to classify the entities mobility patterns; C) *The time scheme matching* which is an assignment of entities with a

particular motion behavior to a suitable timer scheme. Following, we highlight each one of Dribble’s stages deeply.

4.4.1 Entities

Initially, each entity running Dribble starts with a default (pre-set) timer scheme. To set a meaningful default time scheme, we look to the performance and scenarios highlighted on state-of-the-art approaches. Thus, in our experiments, we choose Trickle Timer (TT) to be the default timer scheme because it presents a good trade-off balance [75].

4.4.2 Extraction of Mobility Metrics

By using sensing elements, it is possible to study the entities context and derive useful knowledge from them [16]. Here, we are interested in extract mobility patterns from entities motion by making use of those sensing elements. To do that, the statistical properties can be extracted from data gathered from sensors (e.g., GPS, Accelerometer, Gyroscope, etc.) [61]. Such properties usually are classified as social, spatial, and temporal [6, 31].

Table 4.1 survey different mobility metrics with a short description and their properties type. Social properties, for instance inter-connection time and contact duration, help us to better understand how entities connect each other, how long they stay within each other range, and how they relate to each other. Spatial properties, such as radius of gyration and travel distance, are related to the movement behavior of entities in the space, which can give us intuition concerning coverage area as well as its relationship with locations. Temporal properties, for example visit time and travel distance, can give us insights regarding regularity and rhythm of each entity motion from a location to another, how long they stay in a spot. A detailed review of mobility properties and metrics can be found in [6, 31, 60, 61].

There are several mobility metrics available to process as shown in Table 4.1. From time to time, it is possible to collect those mobility metrics observed from sensing elements data. We recommend performing a feature selection in order to choose metrics which contribute most to the prediction of mobility pattern. This aims to minimize the burden of processing, storage, and power consumption which are usually scarce resources in IoT devices. In our experiments, we use three mobility metrics Travel Distance (TD), Visit Time (VT), and Speed (see Section 4.5 for more details) without degrading the prediction score.

Table 4.1: A non-exhaustive list of mobility metrics, a short description and its classification.

Metric	Short Description	Type
Inter-contact time (INCO) [65]	The time interval between consecutive encounters of a pair of entities.	Social
Contact duration (CODU) [31]	Time of a pair of entities is within each other communication range.	
Encounter regularity (EDGE) [51]	Computes the regularity of a social relationship. Measuring trends of encounters.	
Topological overlap (TOPO) [31]	Represents the overlap of the social relationship between a pair of entities when considering all the encounters.	
Radius of gyration (RADG) [31]	It is the distance of an entity “orbits” around its home location.	Spatial
Travel distance (TRVD) [36]	It is the distance traveled between two consecutive spots.	
Entropy [41]	Quantify the number of times and diversity of visits that a spot receives.	
Visit time (VIST) [60]	It is the time spent by an entity in a location.	Temporal
Travel time (TRVT) [6, 31]	The time that an entity spends while traveling from a location to another.	

4.4.3 Learn-based Model

The next stage of Dribble’s pipeline is to the learning process, through sensed data. As we mentioned before, here the entities mobility context is static or mobile (human-like and non-human). In this sense, we have a classical classification or clustering problem, where we want to figure out in which context the entity is. Although we have tested different models ranging from supervised to unsupervised, we focused on supervised models since we can obtain labeled data in our experiments (see Sec 4.5). We have chosen the Multi-Layer Perceptron (MLP) classifier as learning algorithm [88]. One benefit of MLP is that it can learn a non-linear function for classifying more complex mobility context by projecting the input data into a space where they are linearly separable. In a nutshell, MLP learns a function $f(\cdot) : R^m \rightarrow R^p$, where m is the mobility metrics and p are the mobility patterns. Concerning applicability in IoT devices, our experiments show that a basic MLP architecture can be applied to predict accurately the nodes mobility patterns evaluated.

4.4.4 Timer Scheme Matching

The timer scheme matching is a key stage of Dribble. Traditionally, just one timer scheme governs the entire construction and maintenance of the IoT network, which is a hard-assumption, especially if the network entities present mobility capabilities. However, the main Dribble goal is to set a specific timer scheme for each entity conveniently. This makes sense since entities have different resource capabilities and motion behaviors. Thus, they should have different timer schemes to better accommodate their demands.

Currently, Dribble relies on a specialist to make the matching between mobility patterns and timer schemes. To do that assignment, the specialist needs to understand the mobility pattern and which is the timer scheme more suitable. However, this is a hard task, thus to mitigate this burden, the network specialist can use the simulation methodology applied in Section 4.5 to support its decision.

4.4.5 Motion Event

An important step of Dribble is motion event detection. It is expected that entities change their mobility behavior over time. For example, humans sometimes behave like a static entity (e.g., working for long periods in an office), sometimes as a mobile entity (e.g., moving in a grocery). Dribble can re-evaluate the entity mobility pattern by redoing its process timely when a motion event is detected.

Dribble relies on the device's GPS readings to detect motion events. We can separate into two phases the IoT device movement. The first one, a static phase where the device spends some time staying in the same location/spot. The second one, a motion phase where it moves from a location to another location of interest. To detect motion events, we turn our attention to the static phase, because is in this phase we interpret that devices manifest their intention to change its motion pattern. For instance, the device is moving similarly to a human during its movement phase and then it stops and it behaves like a static (or almost static entity). To extract the static phase from GPS readings, we apply a simple heuristic that depends on two-scale parameters, ΔT_{thresh} a time threshold, and δ_{thresh} a distance threshold. This heuristic is similar to that presented in [135, 136]. Following, we present the heuristic:

Definition 1. GPS raw points: Let $P = \{p_1, p_2, \dots, p_n\}$ be the GPS readings containing the latitude, longitude, and timestamp for each point $p \in P$. Figure 4.6a shows a raw trace of GPS points over timer of an entity moving.

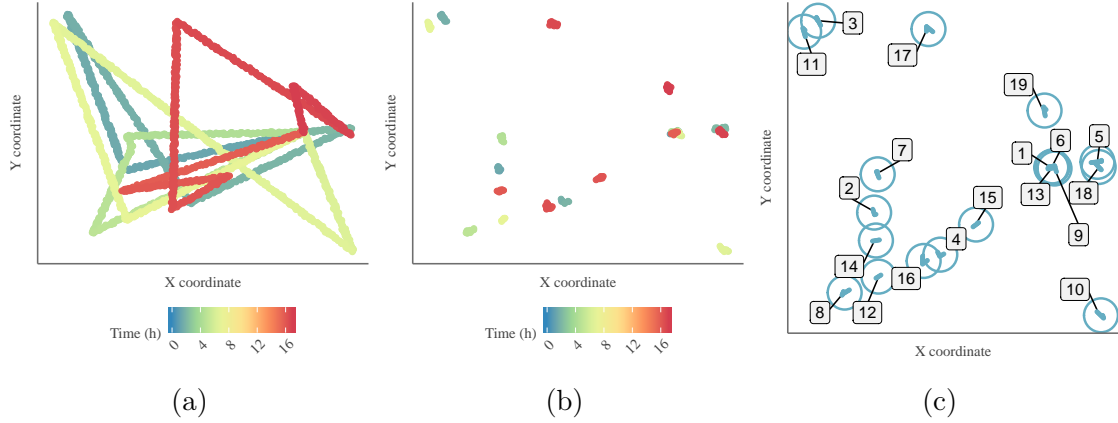


Figure 4.6: (a) a trace of raw GPS points forming a trajectory of an entity. (b) extracted Stay Location from a trajectory (a). (b) Stay location history.

Definition 2. GPS trajectory: We define a device GPS trajectory a sequence of GPS points within a certain time interval. Thus, $Traj = \{p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_n\}$, where $p_i \in P$, $Time(p_{i+1}) > Time(p_i)$, and $Time(p_{i+1}) - Time(p_i) \leq \Delta T$ for $(1 \leq i < n)$.

Definition 3. GPS Stay location: A Stay Location (SL) is a region where the device stayed within a certain period. To extract the stay location, we apply a simple heuristic that depends on ΔT_{thresh} and δ_{thresh} respectively time and distance threshold. Thus, an SL is identified as a sequence of points from a $Traj$ given by $SL_{points} = \{p_s, p_{s+1}, \dots, p_l\}$ for all $s < i \leq l$, that respect the following two conditions. First, the $Distance(p_s, p_l) \leq \delta_{thresh}$, and second, $\Delta T_{thresh} \leq |Time(p_l) - Time(p_s)|$. We can derive the latitude and longitude of an SL simply averaging its points values. Figure 4.6b shows the stay locations extracted from the raw trajectory points. To set a meaningful ΔT_{thresh} and δ_{thresh} , we rely on human walk velocity (≈ 1.4 m/s) and a feasible 1.4s granularity of GPS readings. Thus, we set the threshold values to very low values $\Delta T_{thresh} = 5$ min and $\delta_{thresh} = 5$ m.

Figure 4.6c displays the Stay Location History (SLH) obtained from after applying the SL method. Thus, SLH is a sequence of SL defined as $SLH = \{sl_1, sl_2, \dots, sl_n\}$, where the average times of the stay locations respect $avgTime(sl_m) < avgTime(sl_n)$ for all $s \leq i < l$.

4.5 Evaluation

In this section, we describe Dribble performance evaluation against the baseline Periodic fixed timer scheme, Reverse Trickle scheme as well as the widely used Trickle

Table 4.2: Default simulation parameters

Simulation setup	
Duration	15 days
# of Nodes	200
Base station	1 center
Distribution	Random
Radio range	200 m UDG
DIM	1500 m \times 1500 m
# of random topologies	15
Transmission Model	CC2420-like
Timer schemes	
Trickle and Reverse Trickle	min = 2 s, max = 1024 s
Periodic	90 s

algorithm.

The evaluation process was conducted on Sinalgo simulator [49] and we use the standard RPL as routing protocol [133]. The RPL implementation has the three data traffic patterns (M2P, P2M, and P2P) enabled, storing and no-storing modes as well as hop-count and ETX as Objective Functions. The RPL specification does not define how and when DIS packets should be sent and also how an unreachable parent should be removed from the preferred parent set. Those mechanisms play a significant role to react upon topology changes, especially derived from mobile nodes. In this sense, we implement a simple but not the most efficient alternative. After a node attaches to a parent, it waits for a small acknowledgment from the parent for every DIO sent. Then, if a node is not acknowledged, then it purges the parent from the preferred parent set and enters in a floating DODAG state [133]. Next, it triggers DIS packets try to re-attach to a valid DODAG. In mobile scenarios, authors have recommended that mobile nodes should prioritize static nodes as a parent [25, 92, 106], we also implement this feature.

Table 4.2 lists the default simulation environment parameters. Our simulation ran in random network topologies composed of 200 nodes. In all topologies, one fixed node represents the base station positioned in the center of the field, 50 static nodes were distributed in a grid fashion, representing the infrastructure. Moreover, there are 149 mobile nodes being that 100 present human-like mobility pattern and 49 present non-human patterns. Following, the mobility patterns are described.

Table 4.3: Mobility models parameters

GRM-MIT		CRWP	
Group Duration	720 h	Node Speed (max)	5 m/s
Path time	300 s	T_{pause} (const.)	600 s
<i>Statistical parameters</i>		# Stops	Unif(0, 10)
α_{gmt}	2	PerMobNodes	50%
β_{gmt}	720		
α_{dur}	2		
β_{dur}	720		
α_{size}	2.24		
β_{size}	30		

4.5.1 Modeling the Entities Mobility

We use GRM and CRWP as human and non-human mobility patterns, respectively. Table 4.3 lists the model’s parameters. For GRM, we set the parameters to reproduce MIT real trace [91] behavior. The statistical parameters are from truncated power laws with cut-off where α_* is the power law exponent and β_* the cut-off value: α_{gmt} and β_{gmt} define the group meeting times distribution parameters; α_{dur} and β_{dur} characterize the time that a group of entities will spend together. Finally, α_{size} and β_{size} define which entities will be at each group meeting. For more parameter’s details see [91]. CRWP has four parameters [106]: i) *Speed*: speed which the mobile entity moves; ii) T_{pause} : the amount of time the entity stays in a destination position; iii) *Stops*: the number of stops that the mobile entity does before returning to its original position; iv) *PerMobNodes*: maximum percentage of entities that are out from its initial position in each instant of time.

4.5.2 Measuring Mobility

In our simulation environment, there are static and mobile entities classes. We used mobility metrics as discussed in Sec 4.4.2 to capture the mobility patterns. MOCHA [31] and BonnMotion [6] are freely available tools utilized to extract the mobility metrics from the traces. Figure 4.7 shows three mobility metrics analyzed for one of our simulated topologies. Static nodes present no speed and Travel Distance (TD), however, the Visit Time (VT) is high. While human-like nodes present high variability in Travel Distance (TD) with moderate Visit Time (VT). The non-human nodes present high-speed variability and high values of TD. Visually, it is possible to see three groups in our dataset, and this insight drove us to the learning-based classification method.

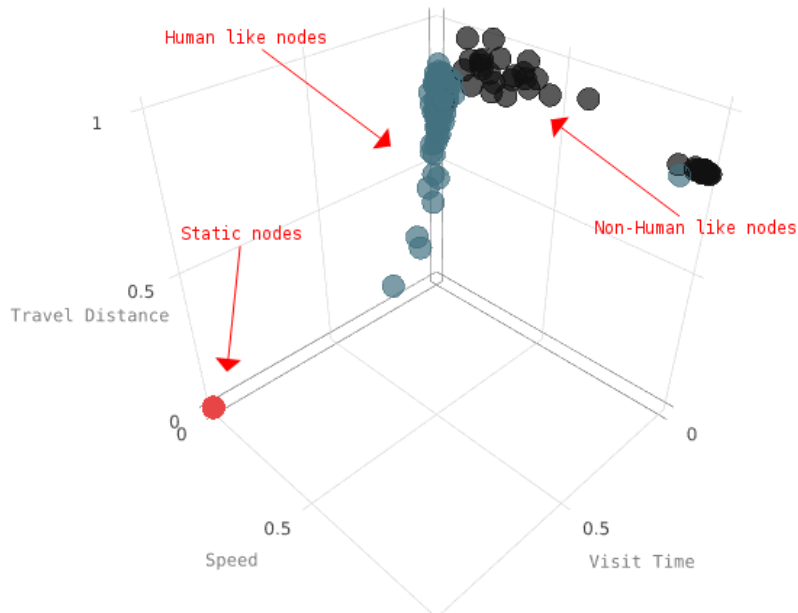


Figure 4.7: Mobility metrics for each entity.

Table 4.4: Model parameters and classification report

Neural Network Architecture and parameters				
Architecture	1 Hidden layer with 100 neurons			
Activation	Rectified linear unit function			
Learning rate	Constant			
# epochs	500			
Weight optimization	Adam			
Train dataset	Mobility metrics from 10 random topologies			
Validation model	10-fold cross-validation			
	Precision	Recall	F1-score	Support
Non-Human	1	0.99	0.99	165
Human	0.98	1	0.99	317
Static	1	0.96	0.98	171
avg / total	0.99	0.99	0.99	653

4.5.3 The Neural Network

We employ a Multi-Layer Perceptron to learning the mobility context classes (static, human-like, and non-human). The MLP architecture (see Table 4.4) was kept simple aiming to be suitable for constrained devices. The model was trained with mobility metrics measured from 10 random topologies composed of 200 nodes each. As model validation, we used 10-fold cross-validation over the data. Moreover, Figure 4.8 shows

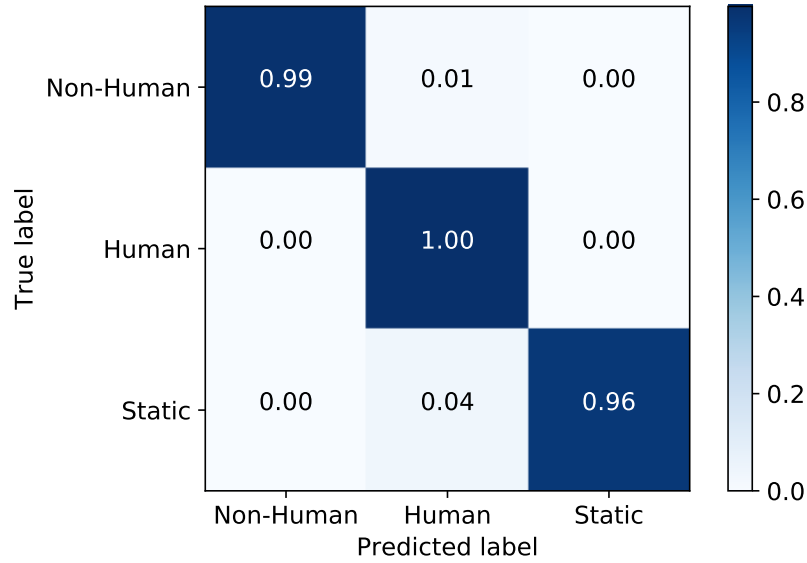


Figure 4.8: Confusion matrix.

the confusion matrix and Table 4.4 lists values of precision, recall, F1-score, and support. The results show high precision and recall, this is due to the target classes present disparate mobility metrics characteristics. Therefore, the model fits and predict correctly the entities classes.

4.5.4 Assigning Timer Schemes to Mobility Patterns

In our experiments, we assign the timer schemes to mobility pattern as follows: i) Non-human mobile entities were assigned to the Periodic scheme since they present the highest speed variability and travel distance. Therefore, a Periodic scheme with suitable short interval can better capture the mobility behavior. ii) Human-like mobile entities were assigned to RevTT. Those entities presented moderate VT and wide variability in TD. This suggests that entities usually arrived at the destination, stop for a while and then move again. This matches with RevTT proposal (Sec 4.3). iii) Static entities were assigned to TT scheme since they represent infrastructure without any mobility, thus TT offers low control overhead when nodes experience network stability which usually occurs for infrastructure devices.

4.5.5 Simulation Results

To compare Dribble against single timer schemes, we use five remaining random network topologies. In each following plot, the bars or points represent the average, and the error bars indicate the confidence interval of 95%.

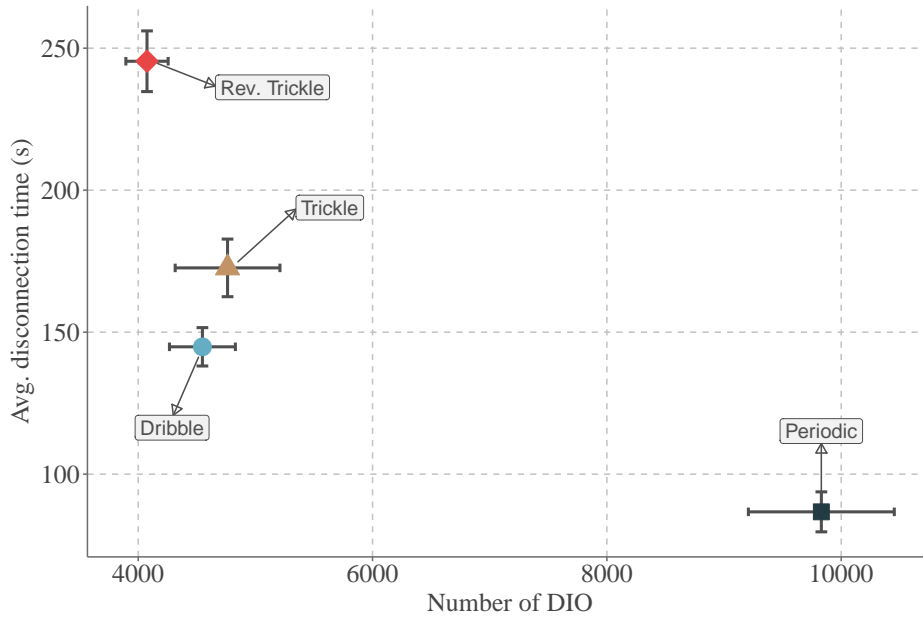
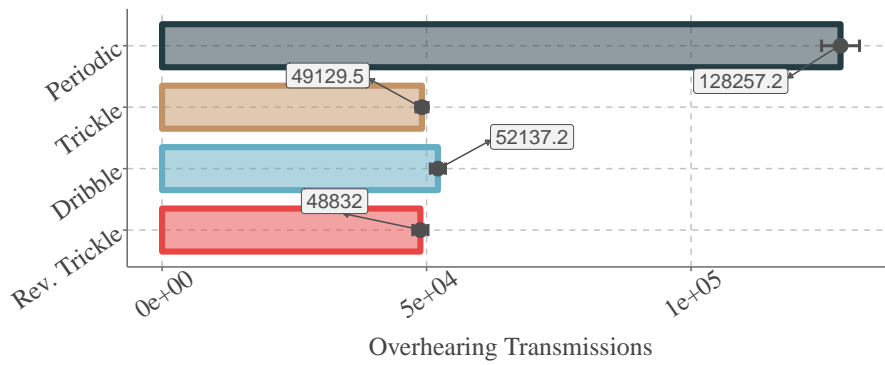
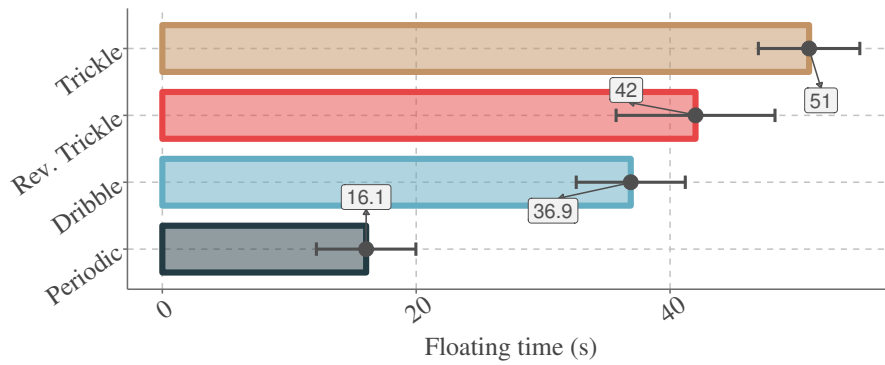


Figure 4.9: The trade-off between control overhead and disconnection time.



(a)



(b)

Figure 4.10: The control flow overhearing and the average time in a floating state.

Firstly, we analyze the trade-off between control advertisement overhead and

the average disconnection time along the 15 days of simulation in Figure 4.9. In the graphic, it is desirable low control overhead and short disconnection timer. A low number of control packets implies less energy expenditure and low channel occupancy. The disconnection time is the time spent from the moment that a node moves out from the parent radio range until it finds out a new parent. As expected, Periodic is the fastest timer scheme to find topology problems but it fires more control packets. On the other extreme, RevTT is more economical in terms of control overhead but topology problems persist for a long time. TT shows moderate trade-off balance. Dribble shows a better trade-off balance by quickly reacting to topology changes and triggering fewer control packets. Which can be explained by the customized timer scheme assignment to each node.

The wireless channel is a shared medium, therefore when a node sends a message nearby neighbors nodes may overhear the transmission, even if the message is not intended to them. Which results in ultimately in unnecessary energy waste, but also channel occupancy. Figure 4.10a shows the overhearing transmissions (for all control flow DIOs, DAOs, and Acks) for each timer scheme. RevTT presented the lowest overhearing average, followed closely by Dribble and TT, while Periodic presented the highest overhearing average.

The average time of a set of nodes in a floating state³ is shown in Figure 4.10b. Dribble has a lower time in a floating state than TT and RevTT due to the mix of timer schemes running concurrently. Therefore, it is more likely to a node receive a control packet and fix the topology inconsistencies. Periodic presented the lowest time in the floating state, however, it uses more control advertisement as shown in Figure 4.9.

Data delivery is a key aspect concerning routing in mobility scenarios due to the high topological dynamics. Figure 4.11 presents the data delivery in six random topologies. Note that RPL with any scheme is able to deliver at least 93.2% of data. But RevTT presented the lowest results ranging from 93.2% to 96.8%. Dribble and TT have presented similar results while the Periodic. This is expected since Dribble is able to change its behavior dynamically setting suitably the timer schemes given the entities mobility behavior, thus Dribble tends to present a mid-term among all timer schemes.

³A grounded DODAG offers connectivity (route towards the border router) to hosts, while a floating DODAG does not. It only provides routes to nodes within the floating DODAG [133].

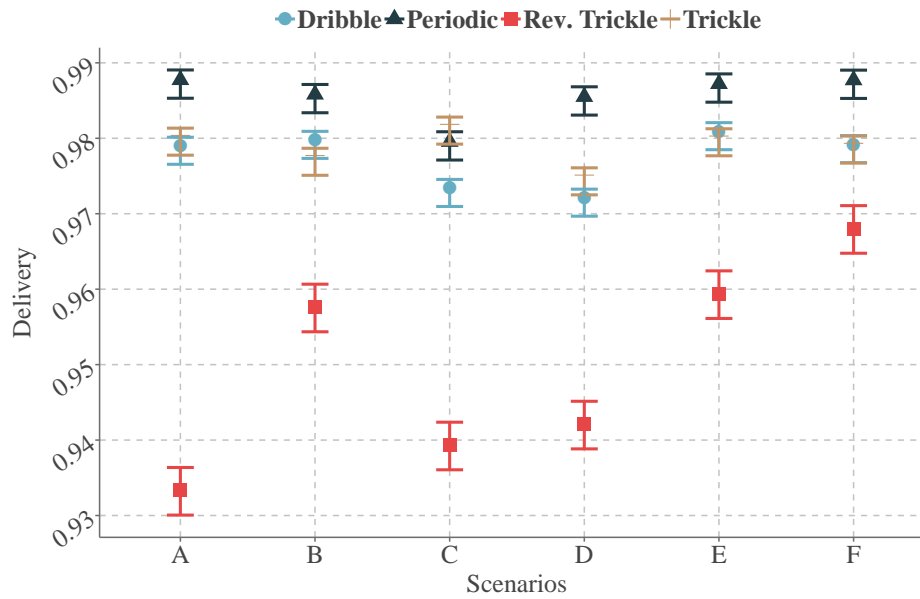


Figure 4.11: The delivery rate for different evaluated scenarios.

4.6 Concluding Remarks

In this work, we proposed Dribble, a learn-based timer scheme selector, to improve the way timer schemes are used in IoT. Until now, routing protocols have used a single timer selector chosen without considering the entities' mobility behavior. Dribble goes further by setting a custom-made timer scheme to proper devices given their mobility pattern. We evaluate Dribble against Trickle Timer and Reverse Trickle Timer. As a result, Dribble presented a better trade-off balance between quick response to topology problems and energy expenditure and channel occupancy.

As future work, we aim to extensively improve Dribble to support fine-grained mobility contexts, better recursive re-evaluation of nodes mobility behavior and beaconing scheme matching. Also, provide an automatic way to associate mobility patterns to timer scheme, avoiding specialist mediation.

Chapter 5

An Alternative Routing Protocol for the Static Internet of Things

5.1 Contextualization

IPv6 over Low-Power Wireless Personal Area Networks ¹ is a working group inspired by the idea that even the smallest low-power devices should be able to run the IP to become part of the IoT. In [86], the authors argue that IoT turns the cities and rural areas into smart environments, where IoT's devices may offer digital services and functions to a variety of groups of users. These devices do interconnections with other devices, often fixed, in the physical infrastructure in the environment (roads, home, cars, body, etc.). The main function of a low-power wireless network is usually data collection. Applications based on data collection are plentiful, examples include environment monitoring [123], field surveillance [126], and scientific observation [130]. To perform data collection, a cycle-free graph structure is typically maintained and a convergecast is implemented on this network topology. Many operating systems for sensor nodes (e.g. Tiny OS [73] and Contiki OS [33]) implement mechanisms (e.g. CTP [48] or the RPL [133]) to maintain cycle-free network topologies to support data-collection applications.

In some situations, however, data flow in the opposite direction – from the root, or the border router, towards the leaves becomes necessary. These situations might arise in network configuration routines, specific data queries, or applications that require reliable data transmissions with acknowledgments. Standard routing protocols for low-power wireless networks, such as CTP and RPL, have two distinctive charac-

¹We use the acronym 6LoWPAN to refer to Low power Wireless Personal Area Networks that use IPv6

teristics: communication devices have unstructured IPv6 addresses that do not reflect the topology of the network (typically derived from their MAC addresses), and routing lacks support for any-to-any communication since it is based on distributed collection tree structures focused on bottom-up data flows (from the leaves to the root).

The specification of RPL defines two modes of operation for top-down data flows: the non-storing mode, which uses source routing, and the storing mode, in which each node maintains a routing table for all possible destinations. This requires $O(n)$ space (where n is the total number of nodes), which is unfeasible for memory-constrained devices. Our experiments show that in random topologies with one hundred nodes, with no link or node failures, RPL succeeds to deliver less than 20% of top-down messages sent by the root (see Figure 5.8).

Some works have addressed this problem from different perspectives [34, 101, 113]. Counting Bloom Filter Routing (CBFR) [101] is a routing scheme that builds upon collection protocols to enable point-to-point communication. Each node in the collection tree stores the addresses of its direct and indirect child nodes using Bloom filters to save memory space. Opportunistic RPL (ORPL) [34] also uses bloom filters and brings opportunistic routing to RPL to decrease control traffic overload. Both protocols suffer from false positives problem, which arises from the use of Bloom filters. Even though CTP does not support any-to-any traffic, XCTP [113], an extension of this protocol, uses opportunistic and reverse-path routing to enable bi-directional communication in CTP. XCTP is efficient in terms of message overload, but exhibits the problem of high memory footprint.

In this work, we build upon the idea of using hierarchical IPv6 address allocation that explores cycle-free network structures and we propose Matrix, a routing scheme for dynamic network topologies and fault-tolerant any-to-any data flows in 6LoWPAN. Matrix assumes that there is an underlying collection tree topology (provided by CTP or RPL, for instance), in which nodes have static locations, i.e., are not mobile, and links are dynamic, i.e., nodes might choose different parents according to link quality dynamics. Therefore, Matrix is an overlay protocol that allows any low-power wireless routing protocol to become part of the Internet of Things. Matrix uses only one-hop information in the routing tables, which makes the protocol scalable to extensive networks. In addition, Matrix implements a local broadcast mechanism to forward messages to the right subtree when node or link failures occur. Local broadcast is activated by a node when it fails to forward a message to the next hop (subtree) in the address hierarchy.

After the network has been initialized and all nodes have received an IPv6 address range, three simultaneous distributed trees are maintained by all nodes: the collection

tree (Ctree), the IPv6 address tree (IPtree), and the reverse collection tree (RCtree). The Ctree is built and maintained by a collection protocol (in our case, CTP). It is a minimum cost tree to nodes that advertise themselves as tree roots. Matrix builds the IPtree over the first stable version of the Ctree in the reverse direction, i.e., nodes in the Ctree receive a hierarchical IPv6 address from root to leaves, originating a static structure. Since the Ctree is dynamic, i.e., links might change due to link qualities, at some point in the execution the IPtree no longer corresponds to the reverse Ctree. Therefore, the RCtree is created to reflect the dynamics of the collection tree in the reverse direction.

Initially, any-to-any packet forwarding is performed using Ctree for bottom-up, and IPtree for top-down data flows. Whenever a node or link fails or Ctree changes, the new link is added in the reverse direction into RCtree, and it remains as long as this topology change persists. Top-down data packets are then forwarded from IPtree to RCtree via a local broadcast. Whenever a node receives a local-broadcast message, it checks whether it knows the subtree of the destination IPv6 address: if yes then the node forwards the packet to the right subtree via RCtree and the packet continues its path in the IPtree until the final destination.

We evaluated the proposed protocol both analytically and by simulation. Even though Matrix is platform-independent, we implemented it as a subroutine of CTP on TinyOS and conducted simulations on TOSSIM. Matrix’s memory footprint at each node is $O(k)$, where k is the number of children at any given moment in time, in contrast to $O(n)$ of RPL, where n is the size of the subtree rooted at each routing node. Furthermore, we show that the probability of a message to be forwarded to the destination node is high, even if a link or node fails, as long as there is a valid path, due to the geometric properties of wireless networks. Simulation results show that, when it comes to any-to-any communication, Matrix presents significant gains in terms of reliability (high any-to-any message delivery) and scalability (presenting a constant, as opposed to linear, memory complexity at each node) at a moderate cost of additional control messages, when compared to other state-of-the-art protocols, such as XCTP and RPL. In addition, when compared to our any-to-any routing scheme, the reverse-path routing is more efficient in terms of control traffic. However, the performance of XCTP is highly dependent on the number of data flows and can be highly degraded when the application requires more flows or the top-down messages are delayed.

To sum up, Matrix achieves the following essential goals that motivated our work:

- **Any-to-any routing:** Matrix enables end-to-end connectivity between hosts located within or outside the 6LoWPAN.

- **Memory efficiency:** Matrix uses compact routing tables and, therefore, is scalable to extensive networks and does not depend on the number of flows in the network.
- **Reliability:** Matrix achieves 99% delivery without end-to-end mechanisms, and delivers $\geq 90\%$ of end-to-end packets when a route exists under challenging network conditions.
- **Communication efficiency:** Matrix uses adaptive beaconing based on the Trickle algorithm [75] to minimize the number of control messages in dynamic network topologies (except with node mobility).
- **Hardware independence:** Matrix does not rely on specific radio chip features, and only assumes an underlying collection tree structure.
- **IoT integration:** Matrix allocates global (and structured) IPv6 addresses to all nodes, which allow nodes to act as destinations integrated into the Internet, contributing to the realization of the Internet of Things.

The rest of this chapter is organized as follows. In Section 5.3, we describe the Matrix protocol design. In Section 5.4, we analyze the message complexity of the protocol. In Section 5.5, we present our analytic and simulation results. In Section 5.2, we discuss some related work. Finally, in Section 5.6, we present the concluding remarks.

5.2 Related Work

Ad hoc On-Demand Distance Vector (AODV) [97] and Dynamic Source Routing Protocol (DSR) [58] are on-demand routing protocols for any-to-any communication. AODV floods the network with messages RREQ to build a path to the destination. On the other hand, the DSR protocol uses the packet header to store the route path. Unlike DSR, our protocol does not store any routing path information in the packet header. The AODV protocol has some similarity with XCTP in the strategy of storing the reverse path. Performing a conceptual comparison between these protocols with Matrix, it is easy to see that Matrix does not save entire routes either in tables or packets. Dymo [17] is the AODV successor. However, it is optimized for MANETs. In the context of low-power and lossy networks, CTP [48] and CodeDrip [59] were designed for bottom-up and top-down data flow, respectively. They support communication in only one direction. CodeDrip is a dissemination protocol that uses network coding to recover lost packets by combining received packets. Our approach is an any-to-any protocol

that also enables dissemination. CTP is an efficient data collection protocol that uses 4-bit [40] metric to estimate the link quality and route cost. Data and control packets are used to obtain the link quality on CTP. MultiHopLQI [131] and MintRoute [134] have the same propose of CTP, but CTP overcomes them as shown in [48]. Centrality-based Green Routing for Low-Power and Lossy Networks (CGR) [112] is a collection routing protocol that considers both centrality and energy to improve network performance and decrease power consumption.

State-of-the-art routing protocols for 6LoWPAN that enable any-to-any communication are RPL [133], XCTP [113], and Hydro [29]. RPL allows two modes of operation (storing and non-storing) for downwards data flow. The non-storing mode is based on source routing, and the storing mode pro-actively maintains an entry in the routing table of every node on the path from the root to each destination, which is not scalable to even moderate-size networks. XCTP is an extension of CTP and is based on a reactive reverse collection route creating between the root and every source node. An entry in the reverse-route table is kept for every data flow at each node on the path between the source and the destination, which is also not scalable in terms of memory footprint. Hydro protocol, like RPL, is based on a DAG (directed acyclic graph) for bottom-up communication. Source nodes need to periodically send reports to the border router, which builds a global view (typically incomplete) of the network topology. More recent protocols [78, 82, 93] modified RPL to include new features. In [93], a load-balance technique is applied over nodes to decrease power consumption. In [78, 82], they provide multipath routing protocols to improve throughput and fault tolerance. Table 5.1 shows a comparison between the 6LoWPANs protocols used in the analysis.

Table 5.1: Comparison between related protocols for 6LoWPAN.

Features	Matrix	RPL	CTP	XCTP
Bottom-up traffic	✓	✓	✓	✓
Top-down traffic	✓	✓		✓
Any-to-any traffic	✓	✓		
Memory efficiency	✓		✓	
Fault tolerance	✓			✓

5.3 Design Overview

The objective of Matrix is to enable an underlying data collection protocol (such as CTP and RPL) to perform any-to-any routing in the IoT network while preserving

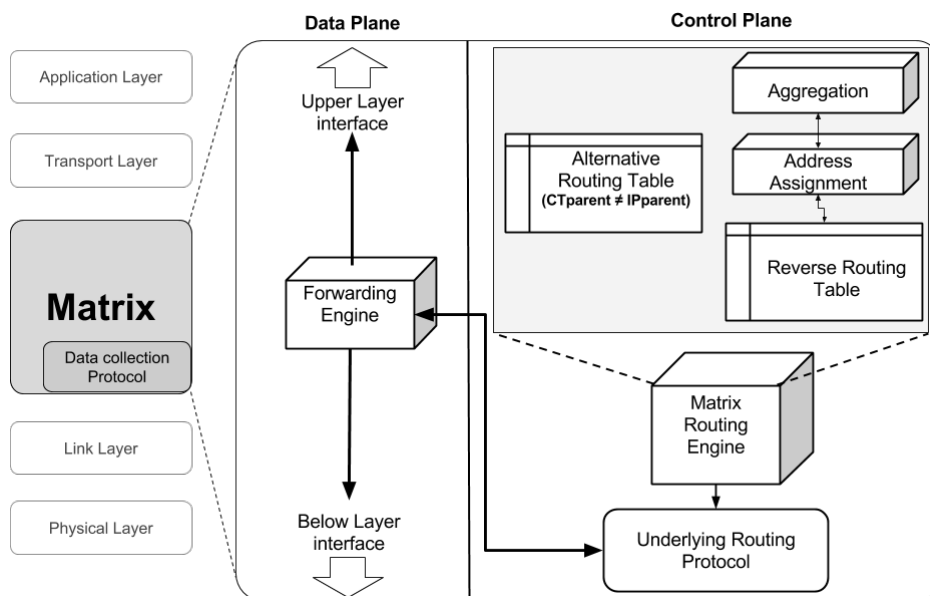


Figure 5.1: Matrix protocol's architecture.

memory and message efficiency, as well as adaptability to networks topology dynamics². Matrix is a network layer protocol that works together with a routing protocol. Figure 5.1 illustrates the protocol's architecture, which is divided into: routing engine and forwarding engine. The routing engine is responsible for the address space partitioning and distribution, as well as routing table maintenance. The forwarding engine is responsible for application packet forwarding.

Matrix encompasses the following execution phases:

1. **Collection tree initialization:** the collection tree (Ctree) is built by the underlying collection protocol; each node achieves a stable knowledge about who its parent is; adaptive beaconing based on Trickle algorithm [75] is used to define stability;
2. **IPv6 multihop host configuration:** once the collection tree is stable, the address hierarchy tree (IPtree) is built using Multihop Host Configuration for 6LoWPAN (MHCL) (Section 5.3.1); this phase also uses adaptive beaconing to handle network dynamics; by the end of this phase, each node has received an IPv6 address range from its parent, and each non-leaf node has partitioned its address space among its children; the resulting address hierarchy is stored in the distributed IPtree, which initially has the same topology as Ctree, but in reverse, top-down, direction.

²Note that Matrix is not designed to address scenarios with node mobility, but only to work with network topology dynamics caused by changes in link quality, as well as node and link failures.

3. **Standard routing:** bottom-up routing is done using the collection tree, Ctree, and top-down routing is done using the address hierarchy represented by the IPtree; any-to-any routing is performed by combining bottom-up forwarding, until the lowest Lowest Common Ancestor (LCA) of sender and receiver, and then top-down forwarding until the destination.
4. **Alternative top-down routing table upkeep:** whenever a node changes its parent in the initial collection tree, it starts sending beacons to its new parent in Ctree, requesting to upkeep an entry in its routing table with its IPv6 range; such new links in Ctree, in reverse direction, comprise the RCtree routing tables for alternative (top-down) routing;
5. **Alternative top-down routing via local broadcast:** whenever a node fails to forward a data packet to the next hop/subtree in the IPtree, it broadcasts the packet to its one-hop neighborhood; upon receiving a local broadcast, all neighbors check if the destination IPv6 belongs to an address range in their RCtree table; if positive, the packet is forwarded to the correct subtree of IPtree. Otherwise, the packet is dropped; we give a geometric argument and show through simulations that such events are rare.

Next, we describe the architecture of Matrix in more detail.

5.3.1 MHCL: Multihop Host Configuration for 6LoWPAN

Matrix is built upon the idea of IPv6 hierarchical address allocation. The address space available to the border router of the 6LoWPAN (e.g., the 64 least-significant bits of the IPv6 address or a compressed 16-bit representation of it) is hierarchically partitioned among nodes connected to the border router through a multihop cycle-free topology (implemented by standard protocols, such as RPL or CTP). Each node receives an address range from its parent and partitions it among its children until all nodes receive an address. Since the address allocation is performed hierarchically, the routing table of each node has k entries, where k is the number of its (direct) children. Each routing table entry aggregates the addresses of all nodes in the subtree rooted at the corresponding child-node. A portion, say $r\%$, of the address space available to each node is left reserved for possible future/delayed connections (parameter r can be configured according to the expected number of newly deployed nodes in the network, see Figure 5.2). We refer to the resulting distributed tree structure as IPtree.

Messages: MHCL uses two message types to build the routing structure: $MHCL_{Aggregation}$ and $MHCL_{Distribution}$ respectively $MHCL_A$ and $MHCL_D$ for short.

Messages $MHCL_A$ are used in the upward routes, from child to parent. This message carries the number of a node's descendants, used in the aggregation phase. Messages of type $MHCL_D$ are sent along downward routes, from parent to child. This message is used for address allocation and contains the address and corresponding address partition assigned to a child node by its parent. Note that the size of the first address and the size of the allocated address partition can have a length predefined by the root, according to the overall address space (we used a value of 16 bits because the compressed host address has 16 bits). This information is sufficient for the child node to decode the message and execute the address allocation procedure for its children.

Algorithm 1: Matrix: Stabilization timer

```

1 parentDefined ← False
2 maxTime ← rand( $\frac{1}{2} \times \text{Trickle}_{\min}$ ,  $\text{Trickle}_{\min}$ )           // Reset timer
3 while not parentDefined do
4     if not root and timer-off then
5         if parent changed then
6             | Reset timer
7         else
8             if timer < maxTime then
9                 | timer ← 2 × timer                       // double timer
10            else
11            | parentDefined ← True

```

Network stabilization: In order to decide how the available address space is partitioned, nodes need to collect information about the topology of the network. Once a *stable* view of the network's topology is achieved, the root starts distributing address ranges downwards to all nodes. Note that the notion of stability is important to implement a coherent address space partition. Therefore, MHCL has an initial set-up phase, during which information about the topology is progressively updated until a (predefined) sufficiently long period goes by without any topology changes. To implement this adaptive approach, we use Trickle-inspired timers to trigger messages (Algorithm 1). In Algorithm 1 two parameters are used: Trickle_{\min} is the minimum time interval used by the Trickle algorithm, and $spChild$ is a multiplication factor used to define the maximum time interval, such that, if no changes occur within it, then the parent choice becomes stable, and the local variable *parentDefined* is set to TRUE. Since Matrix starts running at the same time as the underlying protocol (in our case, CTP), in the initial state of the network nodes do not have any information about

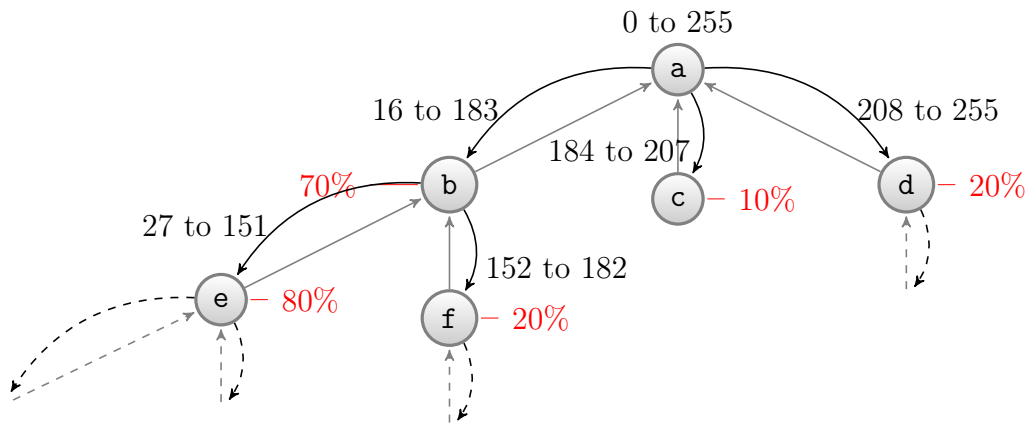


Figure 5.2: MHCL: simplified IPtree example: 8-bit address space at the root and 6.25% reserved for future/delayed connections.

neighboring links. CTP uses 4-bit [40] metric (that uses the expected transmission count, or ETX) to estimate the link quality and route cost. Therefore, Matrix does not know when a node has finally chosen the best connection to its neighbor, i.e., the node with the lowest ETX. That’s why Matrix uses the Trickle timer to define what we call a “stable” network configuration. Note that, once the network reaches an initial state of stability, later changes to topology are expected to be of local nature, caused by a link or a node failure, or a change in the preferred parent of a node. In these cases, the address allocation does not need to be updated, since local mechanisms of message re-transmission can be used to improve message delivery rates, as described in Section 5.3.4.

Descendants convergecast: After the initial network stabilization, each node n_i counts the total number of its descendants, i.e., the size of the subtree rooted at itself, and propagates it to its parent. Moreover, n_i saves the number of descendants of each child. If a node is not the root, and it has defined who the preferred parent is (*parentDefined* is TRUE) it starts by sending a $MHCL_A$ message with *count* = 0 (Algorithm 2). Then it waits for $MHCL_A$ messages from its children, updates the number of descendants of each child, and propagates the updated counter to the parent until its total number of descendants is stable. If a node is the root, then it just updates the number of descendants of each child by receiving $MHCL_A$ messages until its total number of descendants is stable (Algorithm 3). Parameters *spLeaf* and *spRoot* are used to define stabilization criteria in non-root nodes and the root node, respectively. Once the aggregation phase is completed, the root’s local variable *descendantsDefined* is set to TRUE.

Address allocation: Once the root has received the (aggregate) number of de-

Algorithm 2: Matrix: Aggregation timer (non-root nodes)

```

1 maxTimeLeaf ← spLeaf × Tricklemin
2 timer ← rand( $\frac{1}{2}$  × Tricklemin, Tricklemin) // Reset timer
3 count ← 0 // counts descendants through MHCLA messages
4 while NO-MHCLD-FROM-PARENT do // hasn't received IPv6 range
5     if not root and timer-off then
6         if parentDefined and count < 1 then
7             sendTo(MHCLA, parent) // trigger aggregation
8         if COUNT-CHANGED then
9             sendTo(MHCLA, parent)
10            Reset timer
11        else
12            if timer < maxTimerLeaf then
13                timer ← 2 × timer

```

Algorithm 3: Matrix: Aggregation timer (root)

```

1 descendantsDefined ← False
2 maxTimeRoot ← spRoot × Tricklemin
3 timer ← rand( $\frac{1}{2}$  × Tricklemin, Tricklemin) // Reset timer
4 count ← 0 // counts descendants through MHCLA messages
5 while NOT descendantsDefined do
6     if is root and timer-off then
7         if COUNT-CHANGED then
8             Reset timer
9         else
10            if timer < maxTimerRoot then
11                timer ← 2 × timer
12            else
13                descendantsDefined ← True

```

scendants of each child; it splits the available address space into k ranges proportionally to the size of the subtree rooted at each child (see Algorithm 4). Each node n_i repeats the space partitioning procedure upon receiving its address space from the parent and sends the proportional address ranges to the respective children (always reserving $r\%$ for delayed address allocation requests). The idea is to allocate larger portions to larger subtrees, which becomes important in especially large networks because it maximizes the address space utilization. Note that this approach needs information aggregated along multiple hops, which results in a longer set-up phase.

Algorithm 4: Matrix: IPv6 address distribution

```

1 stable ← descendantsDefined or NOT-ROOT
2 if stable and (IS-ROOT or RECEIVED-MHCLD) then
3   partition available address space
4   foreach child  $c_i$  do
5     sendTo(MHCLD,  $c_i$ ) // Send IPv6 ‘range’
6     if NO ack then
7       sendTo(MHCLD,  $c_i$ ) // retransmit

```

Delayed connections: If an address allocation request from a new child node is received after the address space had already been partitioned and assigned, then the address allocation procedure is repeated using the reserved address space. Because of the network stabilization phase and since a node does not know how many descendants it has after the stabilization, we have delayed connections of nodes that are not accounted for during the addressing stage. After the address allocation is complete, each (non-leaf) node stores a routing table for downward traffic, with an entry for each child. Each table entry contains the final address of the address range allocated to the corresponding child, and all table entries are sorted in increasing order of the final address of each range. In this way, message forwarding can be performed in (sub)linear time.

5.3.2 Control Plane: Distributed Tree Structures

After the network is initialized and all nodes have received an IPv6 address range, three simultaneous distributed trees are maintained on all nodes in the 6LoWPAN: **Ctree:** the collection tree, maintained by the underlying collection protocol (CTP/RPL). **IP-tree:** the IPv6 address tree, built during the network initialization phase and kept static afterward, except when new nodes join the network, in which case they receive an IPv6 range from the reserved space of the respective parent node in the collection tree. **RCtree:** the reverse collection tree, reflecting the dynamics of the collection tree in the opposite direction. Initially, IPtree has the same topology as the reverse-collection tree $Ctree^R$, and RCtree has no links (see Figure 5.3a and 5.3b).

$$IPtree = Ctree^R \text{ and } RCtree = \emptyset$$

Whenever a change occurs in one of the links in Ctree, the new link is added in the reverse direction into RCtree and maintained as long as this topology change persists

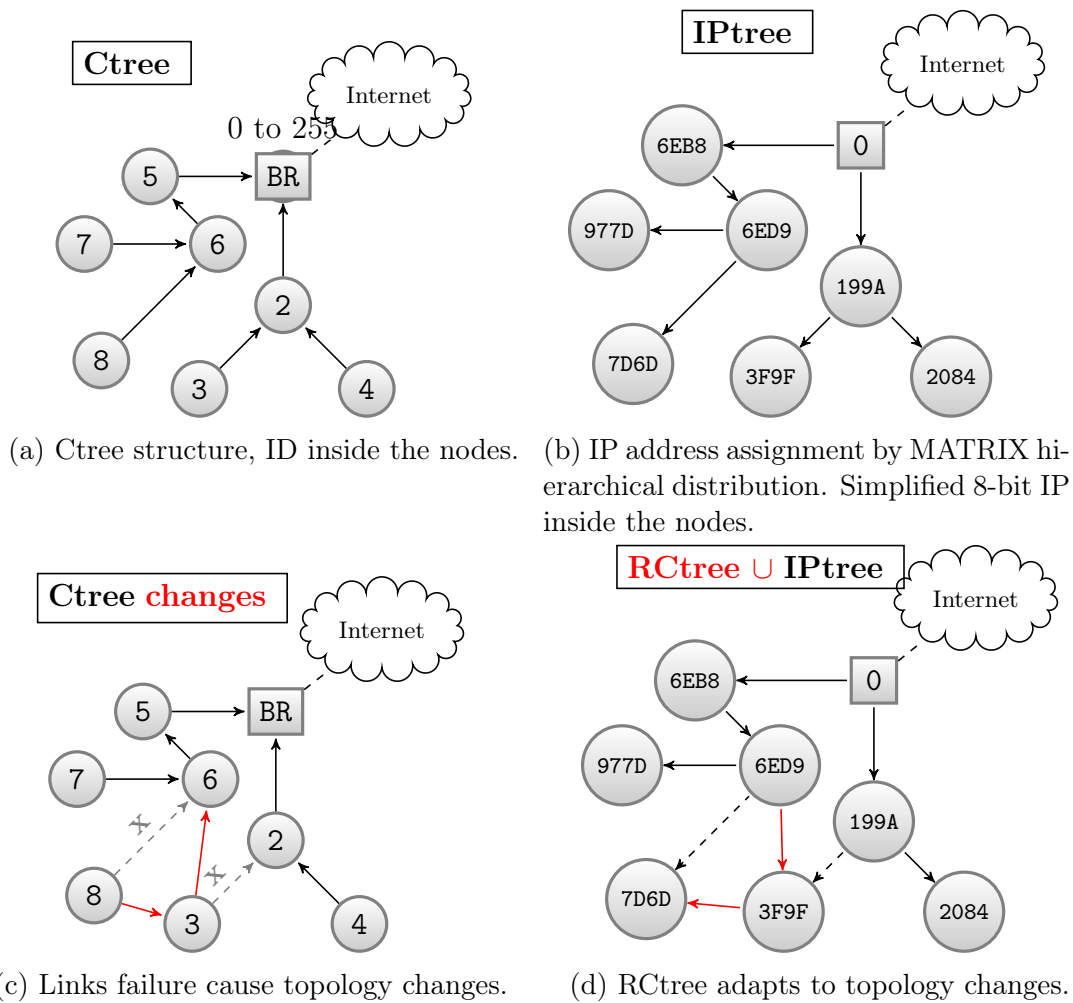


Figure 5.3: RCtree example: before and after two links change in the collection tree.

(see Figures 5.3c and 5.3d).

$$RCtree = Ctree^R \setminus IPtree$$

Therefore, RCtree is not really a tree since it contains only the reversed links present in Ctree but not in IPtree. Nevertheless, its union with the “working” links in IPtree is, in fact, a tree, which is used in the alternative top-down routing:

$$RCtree \cup (IPtree \cap Ctree^R) : \text{alternative routing tree.}$$

Each node n_i maintains the following information:

- $CTparent_i$: the ID of the current parent in the dynamic collection tree;
- $IParent_i$: the ID of the node that assigned n_i its IPv6 range initially $CTarent_i$

= $IParent_i$);

- $IPchildren_i$: the *standard* (top-down) routing table, with address ranges of each one-hop descendant of n_i in the IPtree;
- $RChildren_i$: the *alternative* (top-down) routing table, with address ranges of one-hop descendants in the RCtree.

Note that, each node stores only one-hop neighborhood information, so the memory footprint is $O(k)$, where k is the number of a node's children at any given moment in time, which is optimal, considering that any (optimal) top-down routing mechanism would need at least one routing entry for every (current) child in the tree topology to reach all destinations.

The routing engine (see Figure 5.1) is responsible for creating and maintaining the IPtree and RCtree routing tables. IPtree is created during the network initialization phase, while RCtree is updated dynamically to reflect changes in the network's link qualities. Whenever a node n_i has its $CTparent_i$ updated, and the current parent is different from its $IParent_i$ ($IParent_i \neq CTparent_i$), n_i starts sending periodic beacons to its new parent, with regular intervals (in our experiments, we set the beacon interval to $\delta/8$, where δ is the maximum interval of the Trickle timer used in CTP). Upon receiving a beacon (from a new child in the collection tree), a node ($n_j = CTparent_i$) creates and keeps an entry in its alternative routing table $RChildren_j$ with the IPv6 address range of the subtree of n_i . As soon as n_i stops using n_j as the preferred parent, it stops sending beacons to n_j . If no beacon is received from n_i after $2 \times \delta$ ms, its (alternative) routing entry is deleted. Therefore, links in RCtree are temporary and are deleted when not present in neither the collection nor the IP trees.

5.3.3 Data Plane: Any-to-Any Routing

The forwarding engine (see Figure 5.1) is responsible for application packet forwarding. Any-to-any routing is performed by combining bottom-up forwarding, until the LCA of sender and receiver, and then top-down forwarding until the destination. Upon receiving an application layer packet, each node n_i verifies whether the destination IPv6 address falls within some range $j \in IPchildren_i$: if yes then the packet is forwarded (downwards) to node n_j , otherwise, the packet is forwarded (upwards) to $CTparent_i$. Note that, since each node has an IPv6 address, in contrast to collection protocols, such as CTP and RPL, in Matrix, every node can act as a destination of messages originated inside and outside of the 6LoWPAN.

Each forwarded packet requests an acknowledgment from the next hop and can be retransmitted up to 30 times (similarly to what is done in CTP [48]). If after that no acknowledgment is received, then the node performs a *local broadcast*, looking for an alternative next hop in the RCtree table of a (one-hop) neighbor. The *alternative routing* process is described in detail below.

5.3.4 Fault Tolerance and Network Dynamics

So why is Matrix robust to network dynamics? Note that, since routing is based on the hierarchical address allocation, if a node with the routing entries necessary to locate the next subtree becomes unreachable for longer than approximately one second (failures that last less than 1s are effectively dealt with by retransmission mechanisms available in standard link layer protocols), messages with destinations in that subtree are dropped.

When a node or link fails or changes in Ctree, RCtree reflects this change, and packets are forwarded from IPtree to RCtree via a local broadcast. The node that receives a local-broadcast checks in its RCtree whether it knows the subtree of the destination IPv6 address: if yes then it forwards the packet to the right subtree and the packet continues its path in the IPtree until the final destination.

Consider the following scenario: node X receives a packet with destination IPv6 address D (see Figure 5.4a). After consulting its standard routing table $IP - children_X$, X forwards the packet to C. However, the link $X \Rightarrow C$ fails, for some reason, and C does not reply with an acknowledgment. Then, X makes a constant number (e.g., 30 times in CTP) of retransmission attempts. Meanwhile, since node C also lost its connection to X, it decides to change its parent in the collection tree to node A (see Figure 5.4b). Having changed its parent, C starts sending beacons to A, which creates an entry in its alternative routing table $RC - children_A$ for the subtree rooted at C, and keeps it as long as it receives periodic beacons from C (which will be done as long as $CTparent_C = A$).

Having received no acknowledgment from C, X activates the *local broadcast* mode: it sets the message's type to "LB" and broadcasts it to all its one-hop neighbors (see Figure 5.4c). Upon receiving the local broadcast, node A consults its alternative routing table and finds out that the destination address D falls within the IPv6 address range C. It then forwards the packet to C, from where the packets follow along its standard route in the subtree of C (see Figure 5.4d).

The local broadcast is a reactive mechanism that could be alternatively implemented in a proactive way by adding temporary routing entries to indicate that a link

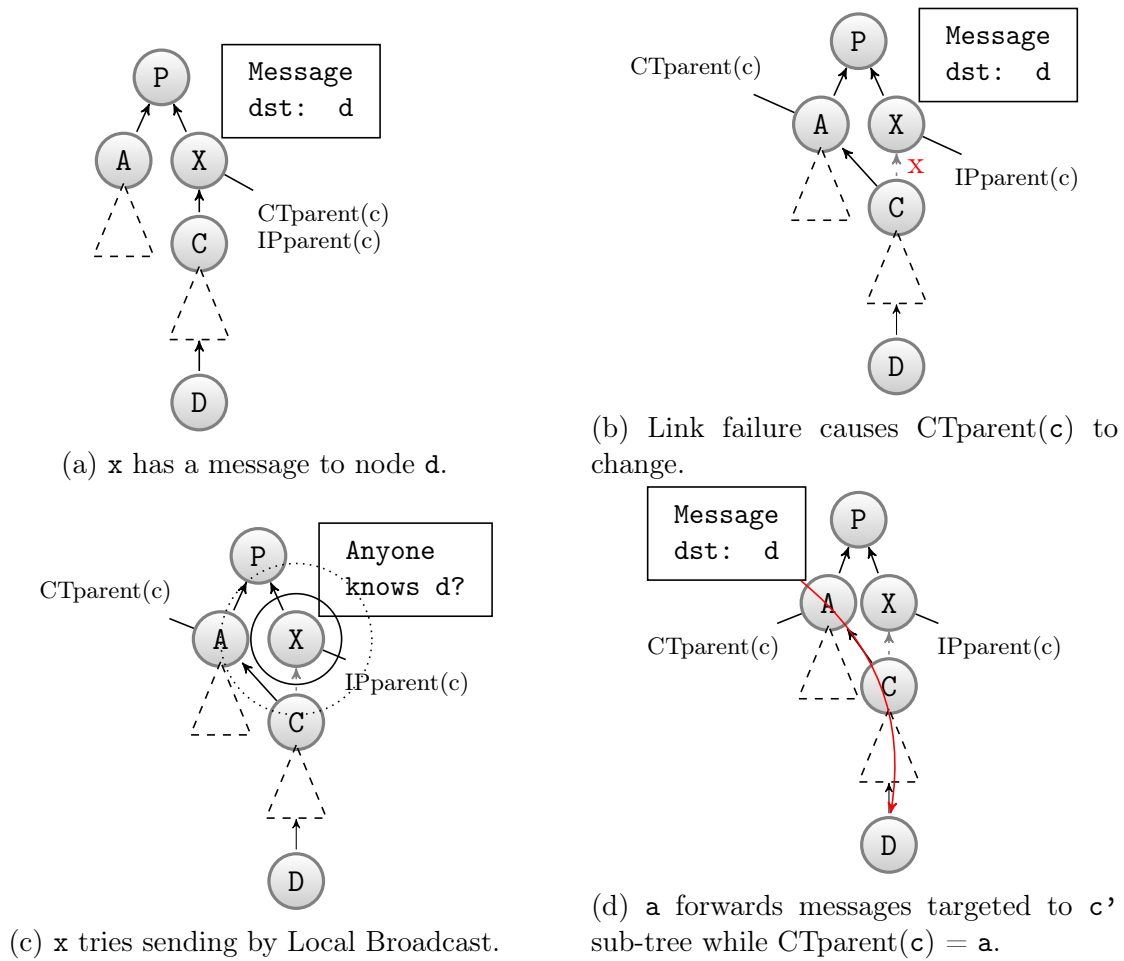


Figure 5.4: Alternative top-down routing upon Ctree change.

has changed to all nodes in the path between the new parent and the LCA. Such a proactive approach could be preferable if Matrix were designed to work in a mobile node environment, where link changes were not local and persistent. However, the memory footprint of such a solution would be linear with the number of link changes in each LCA node's subtree. Local broadcast, on the other hand, handles link dynamics (without node mobility), while guaranteeing a constant memory footprint at each node.

Finally, note that the local broadcast mechanism does not guarantee that the message will be delivered. If no one-hop neighbor of X had the address range of C in its alternative routing table, then the packet would be lost. Nevertheless, we argue that the probability that the message will be forwarded to the appropriate subtree is high.

5.3.5 Alternative Routing: Geometric Rationale

The success of the local broadcast mechanism lies in the ability to forward messages top down along the IPtree, in spite of one or more link or node failures on the way. Note that, whenever a node of IPtree is unavailable, it might not be possible to find the right subtree of the destination. Matrix is designed to handle (non-adjacent) link or node failures and relies on a single local broadcast and temporary reverse collection links (RCtree).

Consider once again the scenario illustrated in Figure 5.4. When a node X is unable to forward a packet to the next hop, it activates the local broadcast mechanism, and it becomes essential that one of X 's one-hop neighbors (in this case A) has replaced X as a parent of C in the collection tree. Therefore, given that the new parent of C is A , it becomes essential that X and A are neighbors. We argue that it is unlikely that this is not the case, and show our argument in a Unit Disk Graph (UDG) model. We use the fact that the number of independent neighbors of any node in a UDG is bounded by a small constant, namely 5 [23].

Given that the maximum number of neighbors that do not know each other is very small, for any possible node distribution and density around X , the probability that two neighbors of X are independent is low. In Figure 5.4c, since both X and A are neighbors of C , the probability that they are themselves neighbors is high. Similar arguments can be used to back the effectiveness of the local broadcast mechanism when dealing with different non-adjacent link and node failures.

Note that this reasoning is only valid in an open space without obstacles and, even then, does not guarantee that the message will be delivered. Nevertheless, our experiments show that this intuition is in fact correct, and Matrix has a 95%–99% message delivery success in scenarios with node failures of increasing frequency and duration.

5.4 Complexity Analysis

In this section, we assume synchronous communication model with point-to-point message passing. In this model, all nodes start executing the algorithm simultaneously and time is divided into synchronous rounds, i.e., when a message is sent from node v to its neighbor u at time-slot t , it must arrive at u before time-slot $t + 1$.

We first analyze the message and time complexity of the IPv6 address allocation phase of Matrix. Then, we look into the message complexity of the control plane of Matrix after the network initialization phase.

Note that Matrix requires that an underlying acyclic topology (Ctree) has been constructed by the network before the address allocation starts, i.e., every node knows who its parent in the Ctree is. Moreover, one of the building blocks of Matrix is the address allocation phase, described in Section 5.3.1.

Theorem 1. *For any network of size n with a spanning collection tree Ctree rooted at node r , the message and time complexity of Matrix protocol in the address allocation phase is $\text{Msg}(\text{Matrix}^{IP}(\text{Ctree}, r)) = \Theta(n)$ and $\text{Time}(\text{Matrix}^{IP}(\text{Ctree}, r)) = \Theta(\text{depth}(\text{Ctree}))$, respectively. This message and time complexity is asymptotically optimal.*

Proof. The address allocation phase of Matrix is comprised of a tree convergecast and a tree broadcast. In the tree convergecast phase, each node sends one message to its preferred parent, informing the size of its subtree, which takes $O(n)$ messages in total and $O(\text{depth}(\text{Ctree}))$ time-slots. In the tree broadcast phase, address range allocation information is distributed from the root to all nodes in the collection tree, also using $O(n)$ messages and $O(\text{depth}(\text{Ctree}))$ time-slots. In the convergecast operation, since every node must send a message to its parent after having received a message from its children, the minimum number of exchanged messages is $\Omega(n)$. Also, a message sent by every leaf node must reach the root, at distance $\leq \text{depth}(\text{Ctree})$, which needs at least $\Omega(\text{depth}(\text{Ctree}))$ time-slots. Similarly, in the broadcast operation, a message must be sent to every node by the respective parent, which needs $\Omega(n)$ messages. Moreover, the message sent by the root must reach every node at $\text{depth}(\text{Ctree})$ hops away, which needs $\Omega(\text{depth}(\text{Ctree}))$ time-slots. Therefore, the message and time complexity of Matrix is asymptotically optimal. \square

Next, we examine the communication cost of the routines involved in the alternative routing, performed in the presence of persistent node and link failures.

Theorem 2. *Consider a network with n nodes and a failure event that causes \mathcal{L}_{CT} links to change in the collection tree Ctree for at most Δ ms. Moreover, consider a beacon interval of δ ms. The control message complexity of Matrix to perform alternative routing is $\text{Msg}(\text{Matrix}^{RC}) = O(n)$.*

Proof. Consider the \mathcal{L}_{CT} link changes in the collection tree Ctree. Note that $\mathcal{L}_{CT} = O(n)$ since Ctree is acyclic and, therefore, has at most $n - 1$ links. Every link that was changed must be inserted in the RCtree table of the respective (new) parent and kept during the interval Δ using regularly sent beacons from the child to the parent. Given a beacon interval of δ , the total number of control messages is bounded by $\Delta/\delta \times \mathcal{L}_{CT} = O(n)$. \square

The analysis presented in this section is based on the synchronous communication model. Nevertheless, in reality, the assumptions of synchronicity and point-to-point reliable message delivery do not typically hold in a 6LoWPAN. The moment in which each node joins the tree varies from node to node, such that nodes closer to the root tend to start executing the address allocation protocol earlier than nodes farther away from the root. Moreover, collisions, node, and link failures can cause delays and prevent messages from being delivered. We analyze the performance of Matrix in an asynchronous model with collisions and transient node and link failures of variable duration through simulations in Section 5.5.

5.5 Evaluation

In this section, we evaluate Matrix performance against state-of-the-art protocols such as RPL [133], CTP [48] and XCTP [113]. In order to do that, we conduct a bulk of experiments through simulation, although Matrix' code is ready to run into real devices. We divide the experiments into three main classes: *memory efficiency*, *protocol overhead*, and *protocol reliability*.

In terms of memory efficiency, we analyze the routing table usage as demand measurement to perform routing, and RAM and ROM footprint as requisites to deploy the protocols. Also, we measured the protocols cost regarding control message overhead to build and maintain routing structures updated, in both dynamic and static scenarios. We also measure the protocol reliability in terms of delivered data packets in both dynamic and static scenarios.

5.5.1 Simulation Setup

Matrix was implemented as a subroutine of CTP in TinyOS [74] and the experiments were run using the TOSSIM simulator [72]. We compare Matrix with and without the local broadcast mechanism, to which we refer as MHCL. XCTP also was implemented in TinyOS. RPL was implemented in ContikiOS [33] and was simulated on Cooja [38].

Firstly, we run the protocols over a static network scenario without link or node failures. Table 5.2 lists the default simulation parameters for the non-faulty scenario. We use the *LinkLayerModel* tool from TinyOS to generate the topology and connectivity model. We also simulated a range of faulty scenarios, based on experimental data collected from TelosB sensor motes, deployed in an outdoor environment [11]. In each scenario, after every 60 s of simulation, each node shutdowns its radio with probability σ and keeps the radio off for a time interval uniformly distributed in $[\varepsilon - 5, \varepsilon + 5]$

Table 5.2: Simulation parameters.

Parameter	Value
Base Station	1 center
Number of Nodes	100
Radio Range (m)	100
Density ($nodes/m^2$)	10
Number of experiments	10
Path Loss Exponent	4.7
Power decay (dB)	55.4
Shadowing Std Dev (dB)	3.2
Simulation duration	20 min
Application messages	10 per node
Max. Routing table size	20 entries

Table 5.3: Faulty network scenarios.

Probability (σ) \ Duration (ε)	Short Dur.	Moderate Dur.	Long Dur.
Low Prob.	(1%, 10 s)	(1%, 20 s)	(1%, 40 s)
Moderate Prob.	(5%, 10 s)	(5%, 20 s)	(5%, 40 s)
High Prob.	(10%, 10 s)	(10%, 20 s)	(10%, 40 s)

seconds. Table 5.3 presents a range of values for A and B, in which A scales from low to high probabilities, and B from short to long time interval. So, each scenario represents a combination of values of σ and ε . Note that these are all node-failure scenarios, which are significantly harsher than models that simulate link or per-packet failures only.

On top of the network layer, we ran two different applications: top-down and any-to-any. In the top-down application, each node sends 10 messages to the root and the root replies with an acknowledgment. In the any-to-any application, each node chooses randomly 10 destination addresses and sends one message to each of those addresses. Nodes start sending application messages 90 s after the simulation has started. The entire simulation takes 20 minutes. Each simulation was run 10 times. In each plot, the curve or bars represent the average, and the error bars the confidence interval of 95%. For each protocol, only results relevant to each plot were included: e.g., CTP does not have a reverse routing table to performs top-down routing, and MHCL differs from Matrix only in faulty scenarios; otherwise, it performs equally and therefore was omitted.

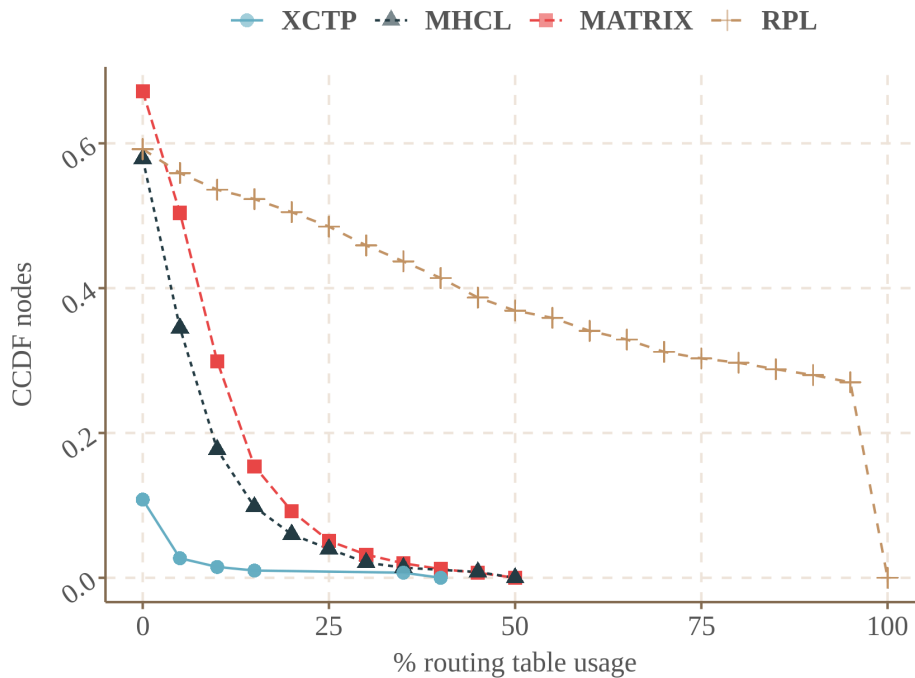


Figure 5.5: Routing table usage CCDF. The routing table size was set to 20 entries.

5.5.2 Results

Firstly, we turn our attention to the memory efficiency of each protocol. To evaluate the use of routing tables, we compare the number of entries utilized by each protocol. Each node was allocated a routing table of maximum size equal to 20 entries. In Figure 5.5, we show the Complementary Cumulative Distribution Functions (CCDFs) of the percentage of routing table usage among nodes³ for Matrix, RPL, XCTP, and MHCL.

In this plot, Matrix was simulated in the faulty scenario, where σ and ε were set to High Probability and Long Duration, respectively (Table 5.3). Note that $> 35\%$ of nodes are leaves, i.e., do not have any descendants in the collection tree topology, and therefore use zero routing table entries.

As we can see, RPL is the only protocol that uses 100% of table entries for some nodes ($\geq 25\%$ of nodes have their tables full). This is because RPL, in the storing mode, pro-actively maintains an entry in the routing table of every node on the path from the root to each destination, which quickly fills the available memory and forces packets to be dropped.

³We measured the routing table usage of each node in one-minute intervals, then took the average over 20 minutes.

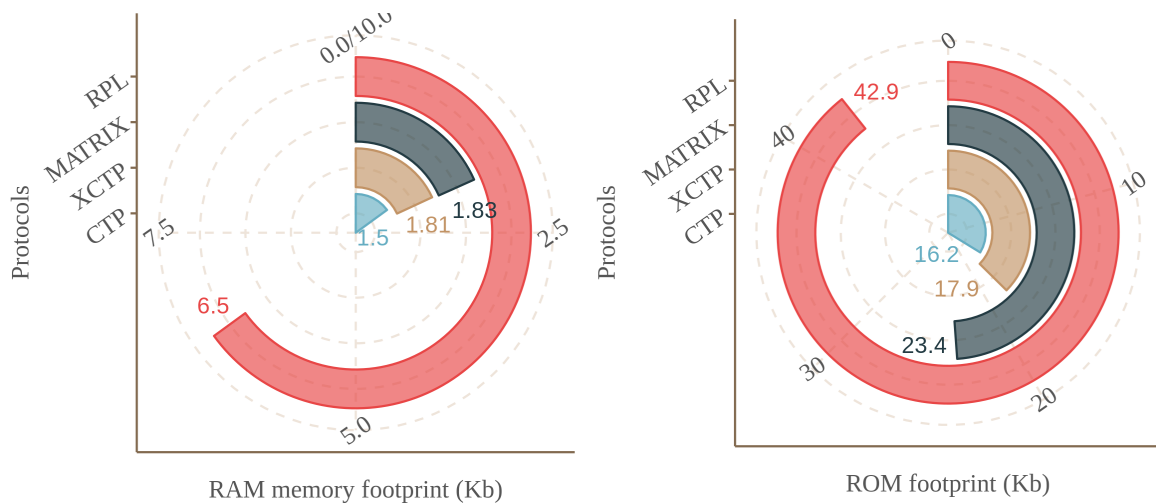
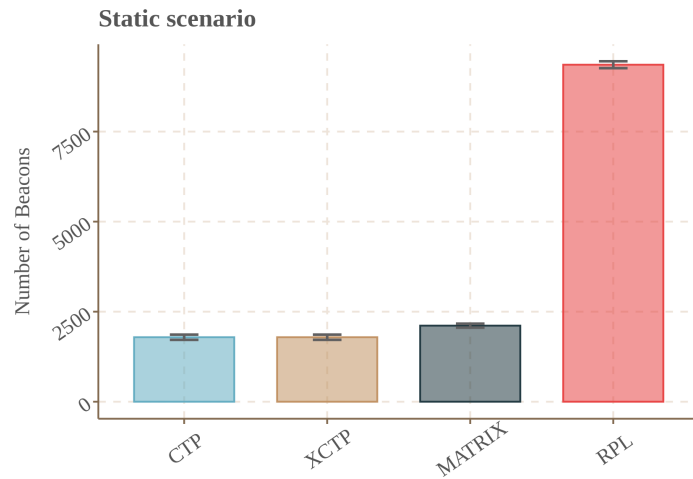


Figure 5.6: Code and memory footprint in bytes.

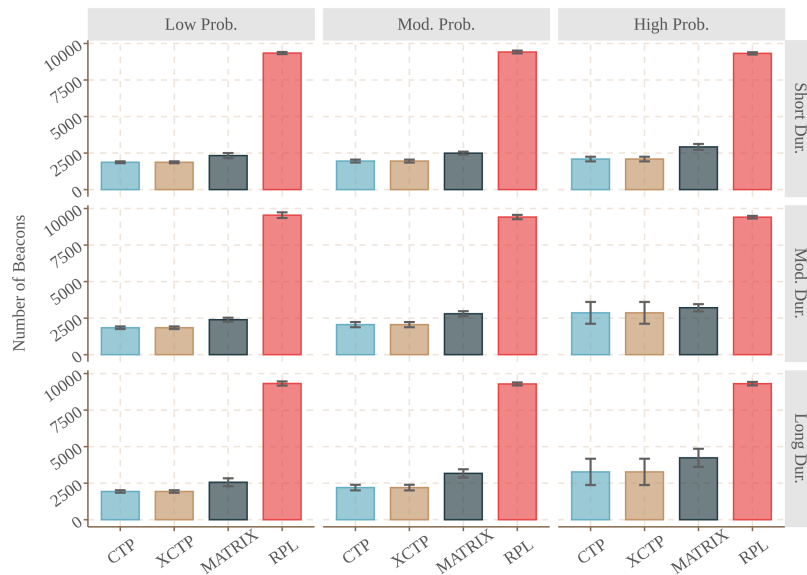
XCTP reactively stores reverse routes only when required. Therefore, the number of routing entries used by XCTP depends on the number of data flows going through each node. Since the simulated flows were widely spaced during the simulation time, the XCTP was able to perform efficiently. The difference between MHCL and Matrix is small: MHCL stores only the IPtree structure, whereas Matrix stores IPtree and RCTree data; the latter are kept only temporarily during parent changes in the collection tree, so its average memory usage is low.

Figure 5.6 compares RAM and ROM footprints in the protocol stack of CTP, RPL, XCTP, and Matrix. We can see that Matrix adds only a little more than 7 Kb of code to CTP, allowing this protocol to perform any-to-any communication with high scalability. When compared with RPL, the execution code of Matrix requires less RAM. Compared to XCTP, Matrix uses almost the same amount of RAM.

In order to evaluate the protocols cost, we measure the protocols overhead to create and maintain the routing structures. Figure 5.7 illustrates the amount of control traffic in our experiments (the total number of beacons sent during the entire simulation). Figure 5.7a shows the protocols cost for the static scenario. Matrix sends fewer control packets than RPL, because it only sends additional beacons during network initialization and in case of collection tree topology updates, whereas RPL has a communication intensive maintenance of downward routes during the entire execution time. Since XCTP is a reactive protocol, it does not send additional control packets, when compared to CTP. Figure 5.7b reports the protocols cost to every combination of faulty parameters. Again, the protocols behavior repeat, but the total amount of control packets increases due to the network dynamics. In the worst scenario case (high



(a) Static scenario.



(b) Faulty scenarios.

Figure 5.7: Number of control packets.

probability and long duration), Matrix presents 45% less control overhead than RPL. Matrix sends 22% more beacons than XCTP and CTP. However, Matrix maintains routes downwards unlike XCTP and CTP.

To evaluate the protocols reliability, we analyze the delivery rate. In Figure 5.8 we compare top-down routing success rate. We measured the total number of application (ack) messages sent downwards and successfully received by the destination.⁴ In the

⁴We do not plot the success rate of bottom-up traffic, since it is done by the underlying collection protocol, without any intervention from Matrix.

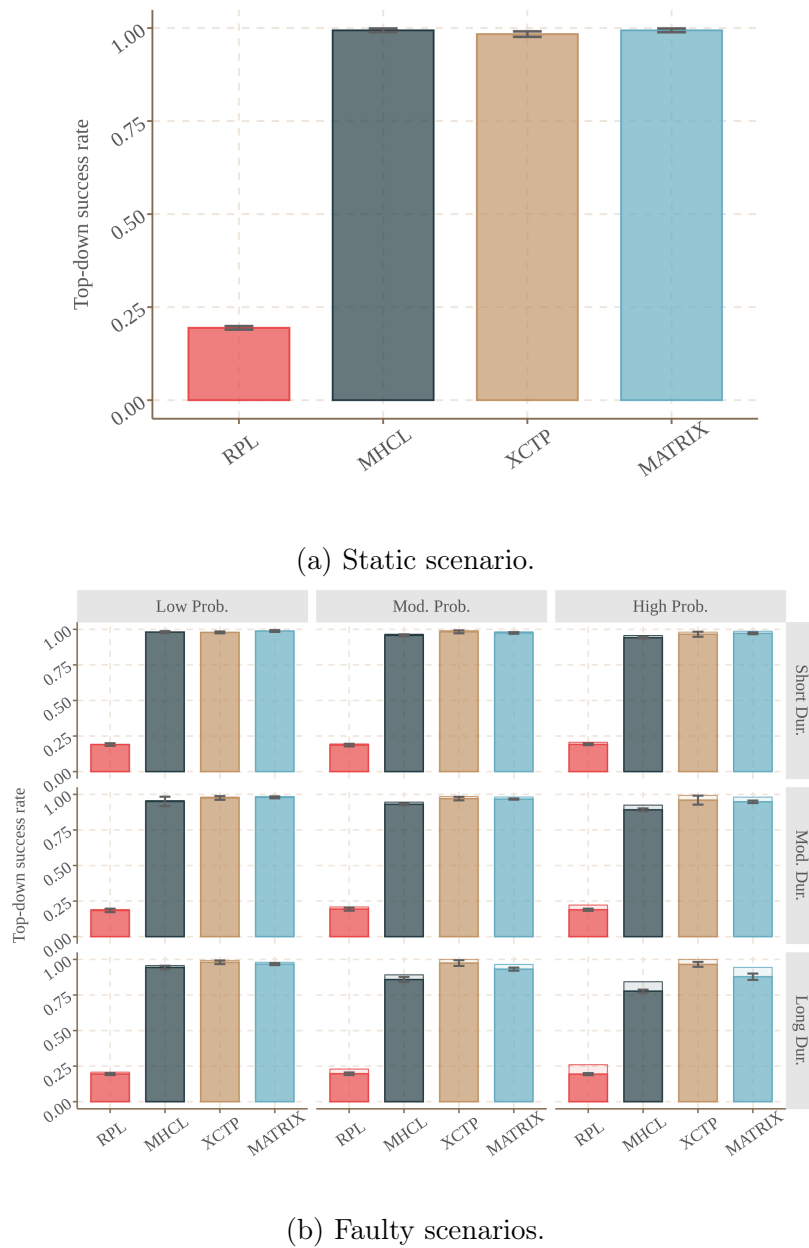


Figure 5.8: Top-down routing success rate.

plot, “inevitable losses” (unfilled bars) refers to the number of messages that were lost due to a failure of the destination node, in which case, there was no valid path to the destination and the packet loss was inevitable. The remaining messages were lost due to wireless collisions and node failures on the packet’s path.

Figure 5.8a shows the protocols top-down success rate for the static scenario. All protocols present high top-down success rate except RPL, which present poor delivery rate. RPL proactively stores entries in the routing table, thus nodes table nearby the root node quickly fill their entries and lack memory to store all top-down routes. In

Figure 5.8b, we present the protocols performance under faulty scenarios. We can see that, when a valid path exists to the destination, the top-down success rate of Matrix varies between 95% and 99%. In the harshest faulty (High Prob. and Long Dur.), without the local broadcast mechanism, MHCL delivers 85% of top-down messages. With the local broadcast activated, the success rate increases to 95%, i.e., roughly 2/3 of otherwise lost messages succeed in reaching the final destination. Note that external factors may be causing RPL's low success rate. Since RPL was the only protocol implemented on Contiki and evaluated in Cooja, native protocols from this OS can interfere with the results. In [14], the authors show how different radio duty cycling mechanisms affect the performance of a RPL network. However, RPL delivered less than 20% of messages in all simulated scenarios due to lack of memory to store routes. Since XCTP is a reactive protocol, it adapts best to failures and dynamics, because downward routes are updated when a message travels upwards. In this way, the top-down success rate of XCTP is higher even in the presence of failures.

In Figure 5.9 we compare the any-to-any success rate. We measured the total number of messages sent by a node that was successfully received by the destination. In this application, each node chooses randomly a destination address and sends a message to this node. We can see that, as expected, there is no significant difference between any-to-any and top-down traffic patterns. Matrix performs any-to-any routing with 90% to 100% success rate, when a valid path exists to the destination. The success rate of RPL remains low, due to lack of memory to store all the routing information needed.

Finally, in Figure 5.10 we compare the response rate of Matrix and XCTP. We calculate the rate of reply by dividing the number of acknowledgments sent by the root by the number of messages received by the root. We vary the reply delay, that is, upon receipt of a message, the root will reply with an acknowledgment after x milliseconds, where $x \in \{ 100, 200, 225, 250, 275, 300, 325, 350, 375, 400, 800 \}$. We can see that the performance of XCTP is highly dependent on the number of data flows. By increasing the application response delay, the number of simultaneous flows increases and the response success rate decreases, because nodes can not store all the information needed. Matrix, on the other hand, does not depend on the number of flows, and the routing table usage is bounded by the number of children of each node.

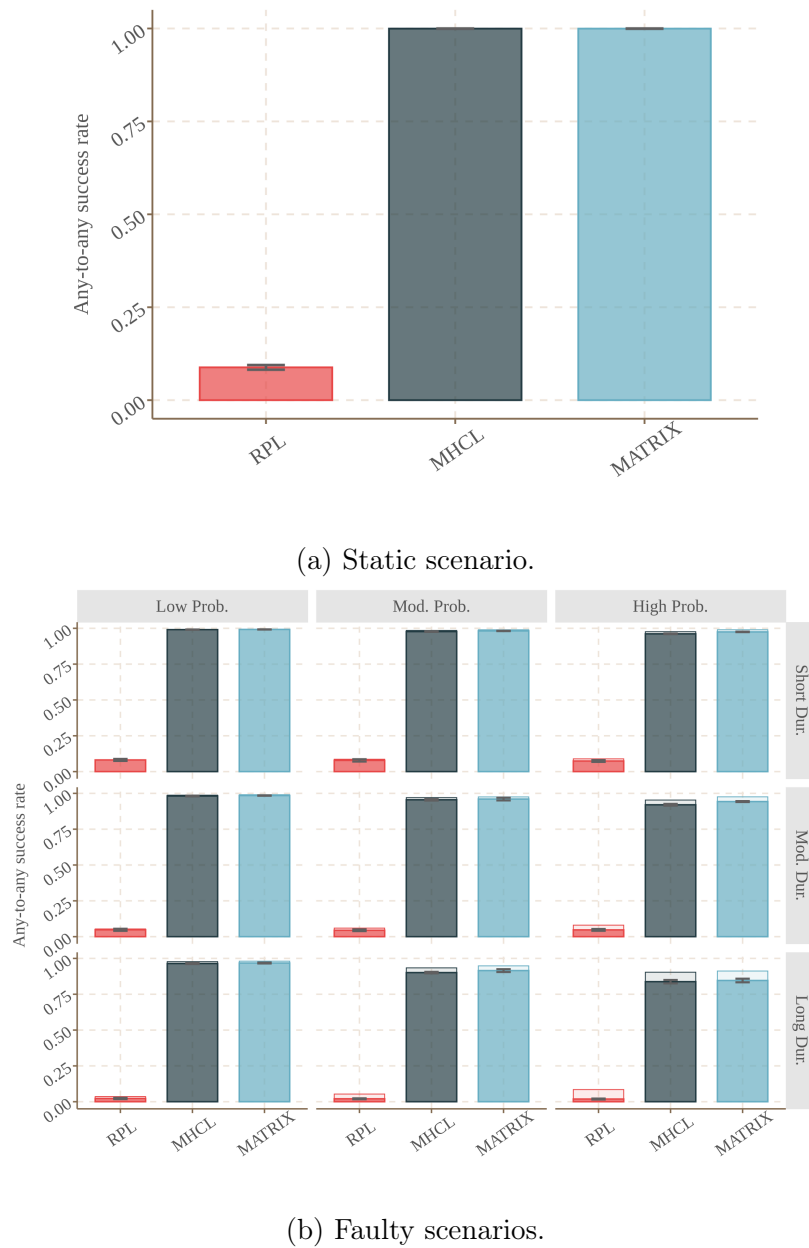


Figure 5.9: Any-to-any routing success rate.

5.6 Concluding Remarks

In this chapter, we proposed Matrix: a novel routing protocol that runs upon a distributed acyclic directed graph structure and is comprised of two main phases: (1) network initialization, in which hierarchical IPv6 addresses, which reflect the topology of the underlying wireless network, are assigned to nodes in a multihop way; and (2) reliable any-to-any communication, which enables message and memory-efficient implementation of a wide range of new applications for 6LoWPAN.

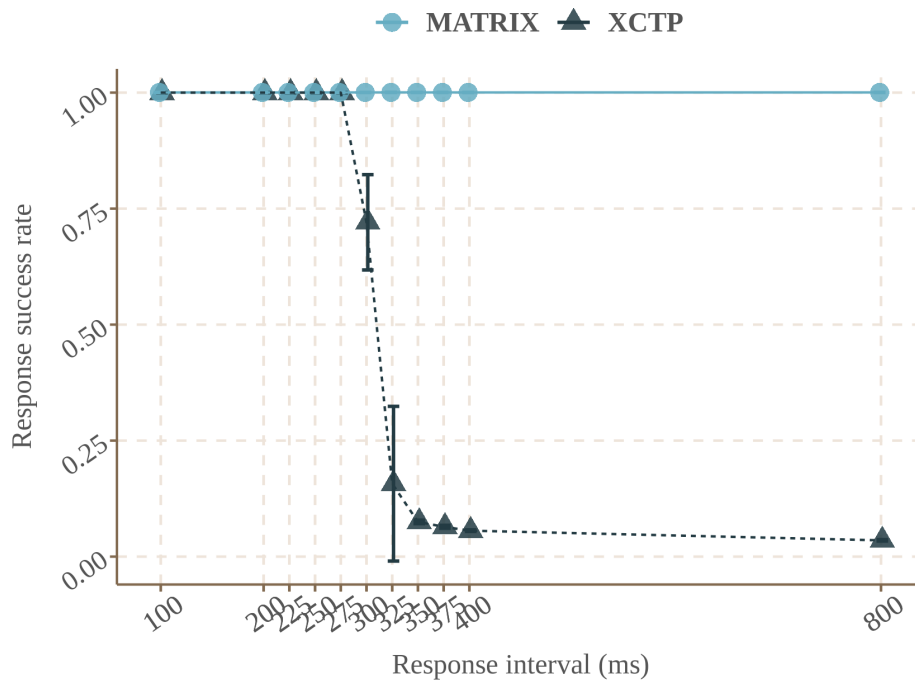


Figure 5.10: Response success rate.

Matrix differs from previous work by providing a reliable and scalable solution for any-to-any routing in 6LoWPAN, both in terms of routing table size and control message overhead. Moreover, it allocates global and structured IPv6 addresses to all nodes, which allow nodes to act as destinations integrated into the Internet, contributing to the realization of the *Internet of Things*.

An interesting future direction is to study mobility in 6LoWPAN. We would like to evaluate the suitability of Matrix in mobile scenarios, where nodes change their point-of-attachment to the 6LoWPAN without changing their IPv6 address, exploring features of the Mobile IPv6 (MIPv6) protocol [96]. Once we consider the mobility of nodes, the local broadcast mechanism that relies on the proximity of the alternative links will no longer hold. This implies that nodes may need to update the routing tables from the new parent to the lowest common ancestor. At first, the memory footprint of such a solution would be linear with the number of mobile descendent nodes (or parent changes) in each node’s subtree, but this will avoid unnecessary retries, timeouts, and broadcast.

Chapter 6

An Alternative Routing Protocol for the Mobile Internet of Things

6.1 Contextualization

IoT has become a reality with the explosive adoption of smart environments, where everyday objects (“things”) are capable of communicating through the Internet. Usually, IoT is a set of interconnected *static* “things” forming a cyber-physical environment. For example, a smart grid composed of smart meters and smart sensors into a smart building. However, mobility is a major factor present in human and non-human life. It makes life easier and turns smart application more flexible and suitable in the mobile world. Nowadays, we already have mobile phones and vehicles in the IoT, in the near future we will have further mobile devices. Naturally, IoT will need to evolve encompassing mobile things (IoMT) and, furthermore, such devices will be able to develop social ties (Social IoT) in the cyber-physical space. With this evolution, IoT takes a step towards ubiquitous computing, where virtually everything is connected with everything at anytime and anywhere.

In the literature, one can find several applications and service proposals for IoMT and SIoT [1, 10, 77, 86]. However, they assume that the networking stack is capable of meeting their requirements of mobility management, memory, energy, and processing efficiency. Actually, we find that we are far from covering all issues imposed by mobility, especially, regarding very tiny devices with resource constraints.

6LoWPAN defines standards for low-power devices that comprise the IoT. Standard routing protocols have been defined, such as CTP [47] and RPL [133], and are widely used to build IoT applications and services. Nonetheless, they do not always

meet the requirements of IoT, IoMT, and SIoT apps and services, such as the mobility management, any-to-any communication, memory efficiency, and others [57].

To address these challenges, we present Mobile Matrix (μ Matrix), a complementary solution to standard routing protocols for IoT, which provides better support for IoMT and SIoT. μ Matrix is a routing protocol that uses hierarchical IPv6 address allocation to manage mobility without changing a node's IPv6 address, while being memory efficient, favoring resource-constrained devices. μ Matrix manages mobility transparently to the upper networking layers in the stack, therefore favoring IoMT and SIoT implementation and adoption. One of the building blocks of μ Matrix is a passive mechanism, named Reverse Trickle Timer (RevTT), to detect mobility in a device's neighborhood. Therefore, the protocol does not need extra hardware to operate. Given that there is an intrinsic trade-off between the delay to detect mobility and the number of control messages, one can tune μ Matrix's frequency of control messages according to the application or the mobility pattern.

μ Matrix is built upon a previously proposed protocol, named Matrix [95]. Besides integrating mobility management into Matrix, μ Matrix presents the following features:

- *Transparent mobility management*: devices can move in the cyber-physical space without ever changing their IPv6 address;
- *Optimal routing path distortion*: messages addressed to a mobile device, from anywhere in the network, are sent along the shortest path from the source to its current location, using its original IPv6 address;
- *Any-to-Any routing*: devices running μ Matrix are able to not just perform bottom-up data collection or top-down dissemination, but also to send messages to any other device in the 6LoWPAN;
- *Passive mobility detection*: μ Matrix uses RevTT to detect neighbor device mobility, which can be tuned according to the application or the mobility pattern;
- *Low memory footprint*: μ Matrix is in consonance with IPv6 addressing and uses a hierarchical address allocation to reduce memory usage to store routing information in a dynamic mobile environment;
- *No fixed devices required*: μ Matrix does not rely on fixed devices to manage mobility, except for the border router, commonly employed in 6LoWPAN;

- *Link dynamics and fault-tolerance support:* μ Matrix inherits from Matrix the capacity to overcome temporary device failure or link dynamics by rerouting the data flows using a local broadcasting mechanism [95];
- *Platform-independent:* μ Matrix does not rely on any specific platform or extra hardware (e.g., GPS, accelerometer) to operate;

Moreover, we propose a new mobility model, to which we refer as *Cyclical Random Waypoint mobility model*. In CRWP, nodes are assigned a home location and might make several moves in random directions, connecting to the 6LoWPAN at different attachment points (and forming social ties), and eventually return to their home locations. Our motivation for proposing a new mobility model comes from application scenarios, where communication is carried out in environments with limited mobility, such as 6LoWPAN deployed in an office or school buildings, university campuses or concert halls or sports stadiums.

The main contributions of this chapter can be summarized as follows:

1. We present μ Matrix, a communication protocol that performs hierarchical IPv6 address allocation and manages routing and mobility without ever changing a node's IPv6 address. The protocol favors the implementation and adoption of IoT, IoMT, and SIoT with constrained devices. μ Matrix has low memory footprint, adjustable control message overhead and achieves optimal routing path distortion;
2. We provide analytic proofs for the computational complexity and efficiency of μ Matrix, as well as an evaluation of the protocol through simulations. We evaluate μ Matrix under human and non-human mobility patterns, therefore showing its functionality;
3. An essential building block of μ Matrix is the RevTT, a passive mobility detection mechanism that captures changes in topology without requiring additional hardware (e.g., accelerometer, GPS or compass).
4. We propose the *Cyclical Random Waypoint (CRWP) model*, a new mobility model for scenarios where devices have a home location and perform periodic random movement patterns.
5. The source code of μ Matrix was made publicly available, so all experimental results presented in this work can be reproduced¹.

¹<https://bps90.github.io/matrix-code/>

Feature	μ Matrix	RPL	Co-RPL	MMRPL	MRPL-V	ME-RPL	mRPL	DRM	XCTP	Hydro
Bottom-up	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Top-Down	✓	✓	✓	✓	✓	✓	✓		✓	✓
Any-to-any	✓	✓	✓	✓	✓	✓	✓			✓
IPv6 support	✓	✓	✓	✓	✓	✓	✓	✓		
Address Allocation	✓									
Memory efficiency	✓									
Fault Tolerance	✓									✓
Local Repair	✓									
Mobility Detection	RevTT	T	P	RTL	P	T	T	T	T	P
Constraints	HL			SN		SN	SN	SN	SN	

Table 6.1: Routing protocol properties (RevTT – Reverse Trickle Timer; RTL – Trever Trickle Timer Like; T – Trickle; P – Periodic; HL – Home Location Assumption; SN – Static Nodes.)

The rest of this chapter is organized as follows. We discuss some related work in Section 6.2. The design overview of μ Mobile is presented in Section 6.3. We analyze the message complexity of the protocol in Section 6.4. We describe the mobility modeling in Section 6.5. In Section 6.6, we present our simulation results. Finally, in Section 6.7, we present the concluding remarks.

6.2 Background and Related Work

WSN are a type of network, where usually tiny static devices are employed to sense, process, store and communicate information surrounding the device. With the integration of Internet Protocol (IP) to WSN emerges the IoT. The concept appeared in early 1982, but its real implementation and adoption started in the last years [129]. More recently, two new paradigms have arisen from IoT: the Social Internet of Things (SIoT) [10] and the Internet of Mobile Things (IoMT) [86]. Those new paradigms hold a common characteristic: their devices are no longer static, but are able to move by itself or are attached to mobile entities.

Several mobility-enabling routing protocols have been proposed for IoT. Table 6.1 summarizes properties of these routing protocols. We use a check mark if the protocol has the feature or empty otherwise. Ten features have been considered, which are related to traffic patterns, addressing, memory, reliability, and protocol limitations.

RPL [133] is a well-known standard routing protocol for 6LoWPAN. Nevertheless, it presents some limitations, in particular in mobility scenarios, such as scalability issues, reliability, and robustness for top-down traffic [57, 95]. Most recent mobile-

enabled routing protocols are RPL extensions [92]. They focus on mobility issues but not always handle the drawbacks of RPL.

Corona RPL (Co-RPL) [43] provides mobility support for RPL but does not take advantage of the dynamic features of the Trickle Timer algorithm, which is intrinsic to RPL. This turns Co-RPL more responsive by using the corona mechanism, but it has a higher overhead. Co-RPL builds on top of RPL's strategies to build and repair downwards routes, which is inefficient regarding memory.

Mobility Management RPL (MMRPL) [25] modifies the RPL beacon periodicity by replacing Trickle with a reverse Trickle-Like mechanism. Thus, their reverse Trickle decays exponentially as long as a node stays attached to the same parent. Mobility Management RPL (MMRPL) assumes that as long as a node remains attached to a parent, the higher will be the probability of the node to move. Also, MMRPL assumes that some nodes are static, which helps its mobility management, therefore requiring heterogeneous devices and more complex code to operate.

Mobile RPL to Vehicular Networks (MRPL-V) [70] modifies RPL to work in vehicular networks. The protocol replaces the Trickle mechanism by a periodic fixed timer to fire RPL control messages. However, since RPL is not built to support highly dynamic scenarios, like vehicular ones, using periodic beaconing to advertise topology changes might be too fast, while the global routing repair is too slow to keep with the changes. Thus, downwards routing can present poor reliability.

Mobility Enhanced RPL (ME-RPL) [37] also assumes that some nodes are static and others can move. In the RPL route discovery phase, when a node is choosing its preferred parent, static nodes have higher priority than mobile ones. When a node is detached from a parent, it sends RPL's DODAG Information Solicitation (DIS) messages in dynamic intervals. Such alterations turn ME-RPL responsive to mobile nodes rejoining the RPL tree and identify mobile nodes. However, the memory requirement to maintain downward routes is still prohibitive.

Mobile RPL (mRPL) [42] is based on the so-called smart-HOP algorithm, a hand-off mechanism for mobile nodes in RPL. They separate nodes into mobile nodes (MN) or serving access points (AP). APs are static and MN nodes use AP nodes as parents in the RPL routing tree.

DAG-based Multipath Routing (DRM) [54] enhances RPL to support data transfer in mobile nodes. DRM removes some features from RPL like top-down and any-to-any traffic handling. DRM also stores routing information to choose the best parent to deliver data reliably.

XCTP [113] extends CTP [47] to support bidirectional traffic and manage mobility. However, XCTP uses a flat address structure instead of IPv6 addressing. Also,

XCTP does not fully support any-to-any routing. Hydro [30] fills the gap of any-to-any communication traffic, but it requires static nodes with a large memory to map nodes' current locations. In fact, XCTP and Hydro were created for static scenarios but can operate under mobility.

MIPv6 [96] supports mobility among different domains by assigning multiple IP addresses to each mobile node: a fixed home address and a Care-of Address (CoA), which changes depending on the current subnet where the node is. A home agent (HA), a fixed entity, is required to manage the mobility and map the addresses. All data traffic is firstly routed to the HA, and then to the CoA, thus MIPv6 does not use the shortest path to routing data flow, presenting a sub-optimal routing path distortion. Hierarchical Mobile IPv6 (HMIPv6) [13] enhances MIPv6 by reducing the signaling load among mobile devices. However, these protocols were not designed for 6LoWPAN, and they do not present a mobility detection mechanism, nor adjustable timers to handle network dynamics.

Protocols for mobile ad hoc networks, like AODV [97] and OLSR [24], have high memory footprint and control message overhead, which makes them not suitable for low-power devices or 6LoWPAN. Actually, there are efforts to extend those for 6LoWPAN such as LOAD [64] and DYMO-Low [63]. These protocols were inspired by AODV and DYMO, but they still present drawbacks in mobile scenarios in terms of memory.

Differently from most of the RPL-based protocols, we present a mobility-enabling routing protocol that deal better with many of RPL's drawbacks. μ Matrix presents low memory footprint for top-down and any-to-any routes under mobility, transparent mobility management, optimal routing path distortion, and fault-tolerance support. Thus, μ Matrix is designed to support IoMT and SIoT implementation and broader adoption.

6.3 Design Overview

As mobility is a new factor to IoT, a question arises: where can we handle mobility in IoT? It is possible to handle mobility with different purposes in different layers of the IoT network protocol stack. However, we argue that mobility in the network layer plays a crucial role in the entire IoT operation in mobile scenarios. Firstly, this is due to the need to maintain routes under device mobility. Secondly, if the device address changes, all protocol layers above the network layer need to be aware of it, increasing the overall protocol complexity. Finally, if devices have constrained resources, then

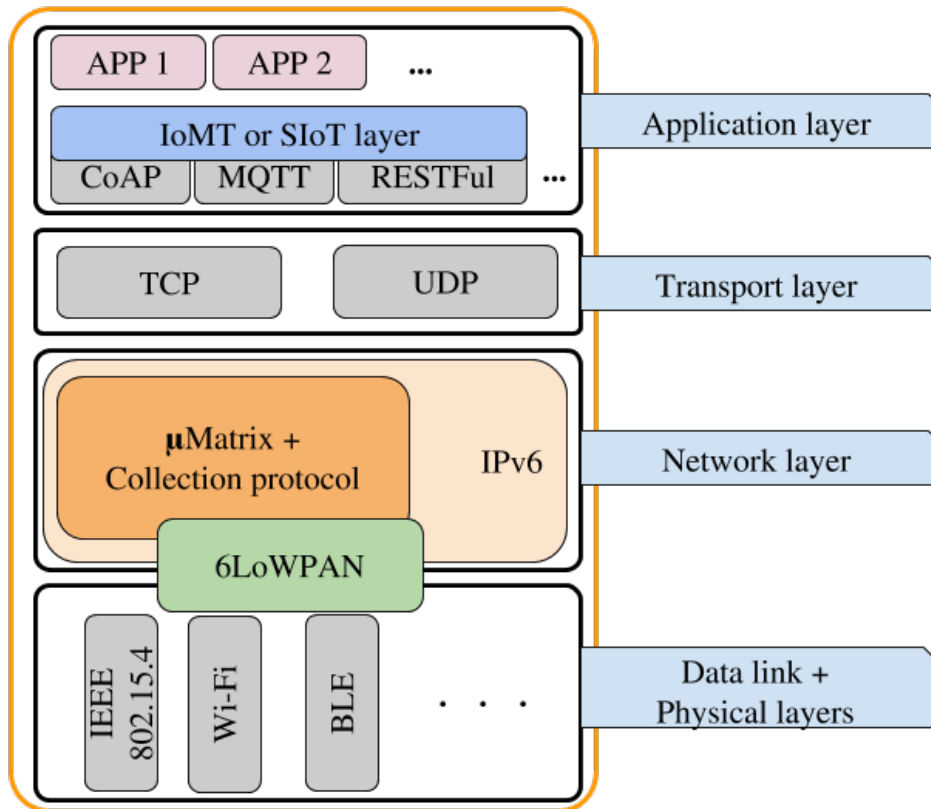


Figure 6.1: μ Matrix integrated into the network stack.

managing routes for mobile devices can be expensive in terms of memory and energy.

There are three data traffic patterns that a routing protocol should provide for IoT, IoMT and SIoT applications [57]:

1. Bottom-up,
2. Top-down,
3. Any-to-any.

The first type of traffic pattern provision is the primary function of standard protocols, such as CTP [47] and RPL [133], and efficient implementations are widely available. However, top-down and any-to-any traffic implementation is not always supported or is not optimized for performance by standard protocols [57]. Therefore, μ Matrix is designed to function on top of an existing bottom-up, or collection, scheme (we use CTP in our implementation), and adds an efficient solution for the remaining two data traffic patterns, besides providing support for mobility.

6.3.1 Mobile Matrix Architecture

Figure 6.1 shows how μ Matrix is integrated into the overall network protocol stack. μ Matrix is implemented in the network layer and is comprised of two planes and sub-modules to handle routing states and manage mobility, as illustrated in Figure 6.2:

- **Control plane:** hierarchical IPv6 address allocation, routing table maintenance, and mobility management;
- **Data plane:** routing information querying and data and control forward data and control packet forwarding.

μ Matrix operates according to the following phases, described in detail in Sections 6.3.2 through 6.3.4:

1. **Collection tree initialization:** an underlying bottom-up routing protocol (e.g., CTP or RPL) creates a collection routing tree, which is dynamically updated to reflect current connectivity conditions;
2. **Hierarchical IPv6 address allocation:** once the collection tree was built, μ Matrix performs a convergecast to gather information about the network's initial topology, which is used to partition the available IPv6 address space in a hierarchical way among all nodes.
3. **Packet forwarding:** bottom-up data traffic is forwarded along the collection tree built and maintained in phase (1); top-down data is forwarded using IPv6 addresses allocated in phase (2); any-to-any packet forwarding is performed by combining the routing structures maintained in phases (1) and (2).
4. **Mobility management:** μ Matrix uses additional routing data structures to reflect the topology changes due to device mobility.

6.3.2 Control Plane: Routing Engine

The control plane manages all routing table structures and makes decisions based on information from other modules, such as mobile or forwarding engines and the underlying collection protocol (Figure 6.2). In this section, we describe the basic routing functionalities of the control plane of μ Matrix, in the following order: routing engine data structure (Section 6.3.2.1), control packets and parameters (Section 6.3.2.2), IPv6 multihop host configuration (Section 6.3.2.3). Then, in Section 6.3.3, we present the mobile engine of μ Matrix.

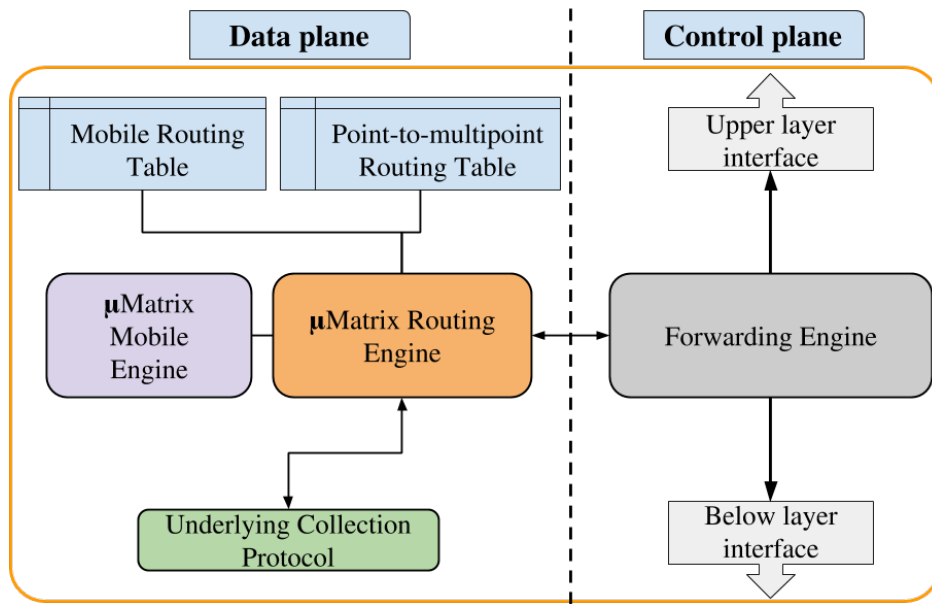


Figure 6.2: μ Matrix protocol's architecture.

6.3.2.1 Routing Engine: Data Structures

μ Matrix and the underlying collection protocol build and maintain three routing trees structures:

1. **Ctree**: a collection tree built by the underlying collection protocol (e.g., CTP [47] or RPL [133]). CTP was used as a data collection routing protocol providing efficiently bottom-up routes. We chose CTP purposely due to its lower code complexity than RPL's [95], robustness, loop avoidance, and its wide acceptance by the community.
2. **IPtree**: an IPv6 hierarchical tree is created using MHCL algorithm [95] at μ Matrix initialization phase. The IPtree is kept static, except when new devices join the network²;
3. **RCtree**: a tree that reflects the topology changes caused by devices mobility.

Figure 6.3 shows those routing structures graphically. Initially, in Figure 6.3a, the underlying collection protocol builds the Ctree structure, note that we have a border router (node 1) that starts the process. Then, μ Matrix starts the MHCL algorithm to distribute the available range of IPv6 hierarchically³. At this moment, IPtree =

²As described in detail in [95], each node is assigned a reserve address space for nodes joining the network after the initialization phase.

³The network operator defines this range of IPs, which will be distributed along the IPtree [95].

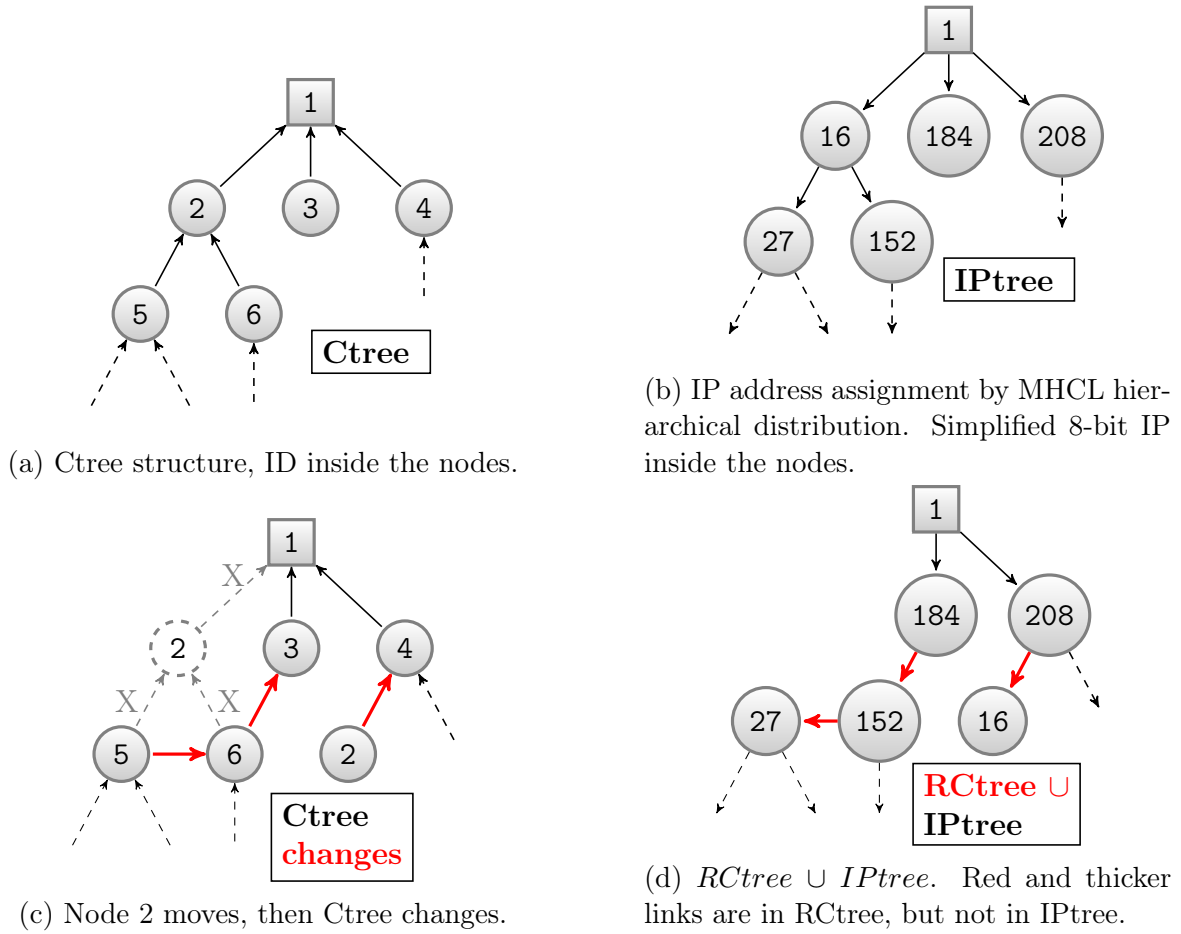


Figure 6.3: Routing structures: Ctree, IPtree, and RCtree.

$Ctree^R$ and $RCtree = \emptyset$ (see Figure 6.3b). Whenever a topology change occurs due to mobility in Ctree, a new reverse link is added into RCtree, and it is maintained while the change remains, therefore $RCtree = Ctree^R \setminus IPtree$ (see Figures 6.3(c)(d)). RCtree is not essentially a tree since it contains only reversed links in Ctree but not in IPtree. Nevertheless, $RCtree \cup IPtree$ is, in fact, a tree, which μ Matrix uses to downward routing. Each node η keeps the following information to build and maintain those trees:

- $CTparent(\eta)$: the ID of the current parent of a node η in the Ctree;
- $PRVparent(\eta)$: the ID of η 's previous $CTparent(\eta)$;
- $IPparent(\eta)$: the ID of the node that assigned to η its IPv6 and IP range;
- $IPchildren(\eta)$: the standard (top-down) routing table with IPv6 ranges for one-hop descendants of η in the IPtree;

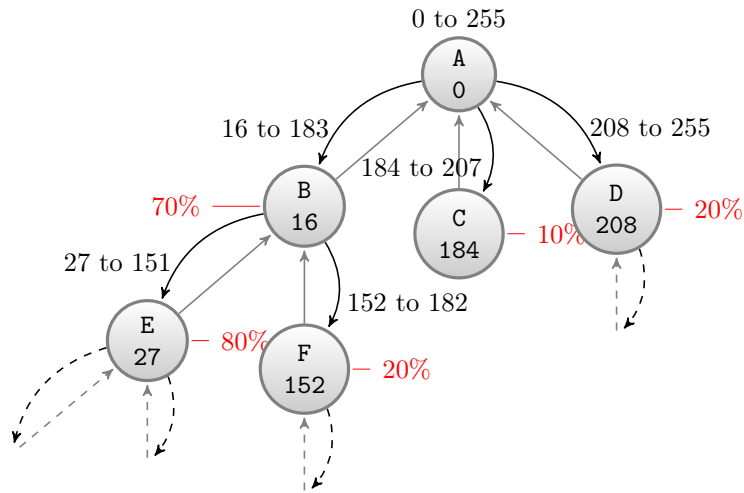


Figure 6.4: Simplified hierarchical address assignment with 8-bit available address space and 6.25% of addresses reserved for delayed nodes. Inside the nodes, its label and IP assigned, the % next to the nodes express the approximate sub-tree size. Thick downwards arrows indicate the available IP range fairly distributed.

- $Mtable(\eta)$: a temporary alternative routing table for mobility management. Each $Mtable$ entry has the following fields: IPv6 range, next hop, and Time Has Lived (THL).

6.3.2.2 Control Packets and Parameters

μ Matrix introduces one new control packet and one parameter to exchange topology information and update the $Mtables$ upon mobility events:

1. **nodeInfo** frame has 7 fields: seqNum, IP Node, IP range, IPparent, CTparent, TTL, and Type. The fields are self-explanatory, except by the type field, which specifies if the frame is a **keepRoute** to an entry into the $Mtable$, or **rmRoute** to remove an entry. Following, we will use **keepRoute** or **rmRoute** for short;
 - A node sends **keepRoute** beacons to inform its current location and the in-network nodes can update its $Mtables$ to reflect the present network topology;
 - **rmRoutes** is an optional beacon. It is sent when nodes move from a location to another in order to quickly remove inconsistent states in-network nodes.
2. δ parameter specifies the time between sending two consecutive **nodeInfo** packets. Note others time-based strategies can be used than the periodic one.

A device fills its *Mtable* by receiving `keepRoute` beacons from mobile nodes. The node keeps *Mtable* entries as long as it receives `keepRoute` (see Section 6.4.1 for *Mtable* memory footprint analysis). Otherwise, it uses a *THL*-based mechanism or `rmRoutes` to remove entries. In static scenarios any node stores one-hop neighborhood information in *IPparent*(η) table, this requires $O(k)$ entries, where k is the number of direct children of a node in the Ctree. This memory footprint is better than state-of-the-art, e.g., RPL would need at least 1 routing entry for every child in a node sub-tree to perform top-down routing.

6.3.2.3 IPv6 Multihop Host Configuration

μ Matrix relies on an underlying collection routing protocol to build the Ctree. Once the Ctree is stable⁴, the address space available to the border router, e.g., the 64 least-significant bits of the IPv6 address (or its compressed 16-bit representation), is hierarchically partitioned among nodes in the Ctree by using the MHCL algorithm [95]. The (top-down) address distribution is preceded by a (bottom-up) convergecast phase, in which each node counts the total number of its descendants and propagates it to its parent. Thus, a node knows how many descendants each child has. Such information is required to distribute IP ranges in a fairly way. As result of this procedure is obtained the IPtree.

Figure 6.4 shows the IP host configuration process. First, the underlying routing protocol creates the Ctree (upwards grey arrows), and then, after the Ctree stabilization, the convergecast phase occurs allowing nodes to know the size of their sub-tree (% next to each node). Next, the border router starts the IP distribution by auto-setting its IP (e.g., the first available IP from range) and then reserving a portion of the range for delayed nodes. After that, the node distributes the remaining range fairly among its children (e.g., in Figure 6.4 B receives 70% of the available range, i.e., from 16 to 183). Finally, each node repeats the IP distribution process.

6.3.3 Control Plane: Mobile Engine

The mobile engine plays a central role in the μ Matrix operation. It is responsible for identifying when mobility events happen and feed the routing engine with current node status. With this, it helps the routing engine to take action upon mobility events properly.

⁴A node is stable in the Ctree if it reaches k times the maximum maintenance beacon period of Ctree protocol without changing its parent. We use Trickle Timer [75] as beacon scheme.

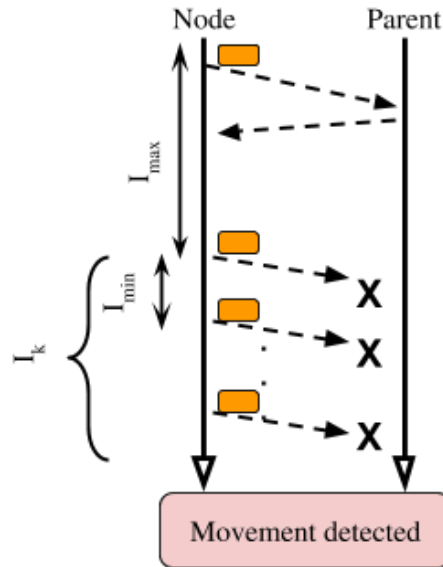


Figure 6.5: Reverse Trickle Timer operation with μ Matrix.

Following, in Section 6.3.3.1, we present details of RevTT a passive mobility detection algorithm. Next, in Section 6.3.3.2, we present the μ Matrix finite state machine used to keep track of the node's status. Finally, in Section 6.3.3.3, we present the μ Matrix strategies to prevent and recover from the loop.

6.3.3.1 Mobility Detection

Mobility detection is a crucial issue to handle mobile devices in routing protocols. Most of the related protocols (see Section 6.2) modify this component to improve the routing protocol performance under device mobility. There are two classes of mobility detection events: i) active mobility event; ii) passive motion event. In the first one, the devices by using extra hardware (e.g., accelerometer or GPS) inform their motion to the routing protocol to take actions. In the second one, the protocol, by itself, infers the movement of the devices (e.g., by using beacons mechanisms).

Standards 6LoWPAN routing protocols, such as CTP or RPL, make use of Trickle Timer algorithm [75] that passively detects topology changes. However, Trickle Timer lacks in agility to detect changes in dynamic network and mobile nodes. We propose a Reverse Trickle Timer (RevTT) that operates similarly to the standard algorithm, but in reverse order.

RevTT introduces three parameters which the network operator must tune according to the application and the mobility pattern requirements. Also, RevTT has three methods to manipulate its behavior. The parameters and methods are described

below:

1. **Parameters:** I_{max} and I_{min} the maximum and minimum time interval to fire an event. I_k the number of attempts before declaring an inconsistency.
2. **Methods:** *Start* aiming to start the RevTT operation, *Stop* aiming to break the timers, and *Reset* that basically stops the current times using *Stops* method and then the *Start* method.

RevTT operation is straightforward. The algorithm starts with I_{max} interval to fire an event. If *Stop/Reset* methods are not used, then RevTT goes to the I_{min} interval for I_k attempts. If nothing happens during I_k fires, then RevTT triggers an event indicating inconsistency.

Figure 6.5 shows the RevTT procedure within μ Matrix Mobile Engine. First, a node starts sending unicast **hasMoved** beacons to its parent in I_{max} intervals. If the node did not receive an acknowledgment for a **hasMoved** beacon, then RevTT sets the interval to I_{min} . After I_k unsuccessful attempts, the node knows that a movement or fault happens. Thus, the node can take actions, for example, properly perform a handover to another parent and then the procedure restarts. Note that by setting the RevTT parameters, the network operator should consider the trade-off between delay to detect mobility and number of beacons. For a smaller delay in mobility detection, I_{max} must be tuned to small values at the cost of more **hasMoved** beacons. In our experiments (see Section 6.6), RevTT parameters were set according to the application data rate.

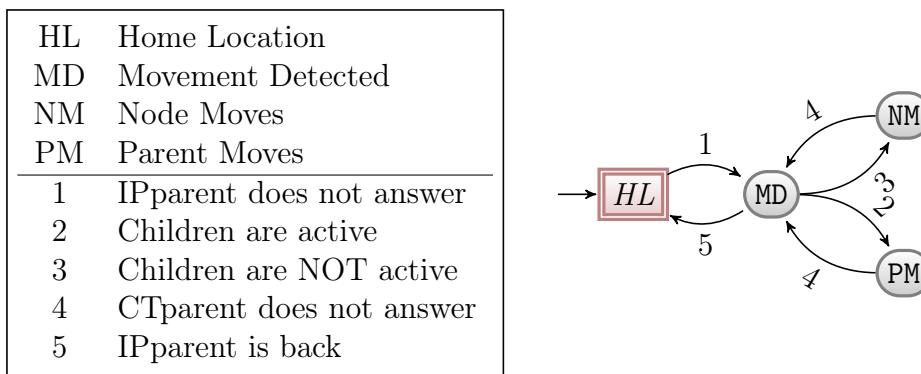


Figure 6.6: Mobile Engine state machine.

In [92], the authors argue that a common modification to support mobility is to change the control message periodicity. The typical approach uses a simple periodic timer or the standardized Trickle Timer. While RevTT waits for $I_{max} + T_k \times I_{min}$ to

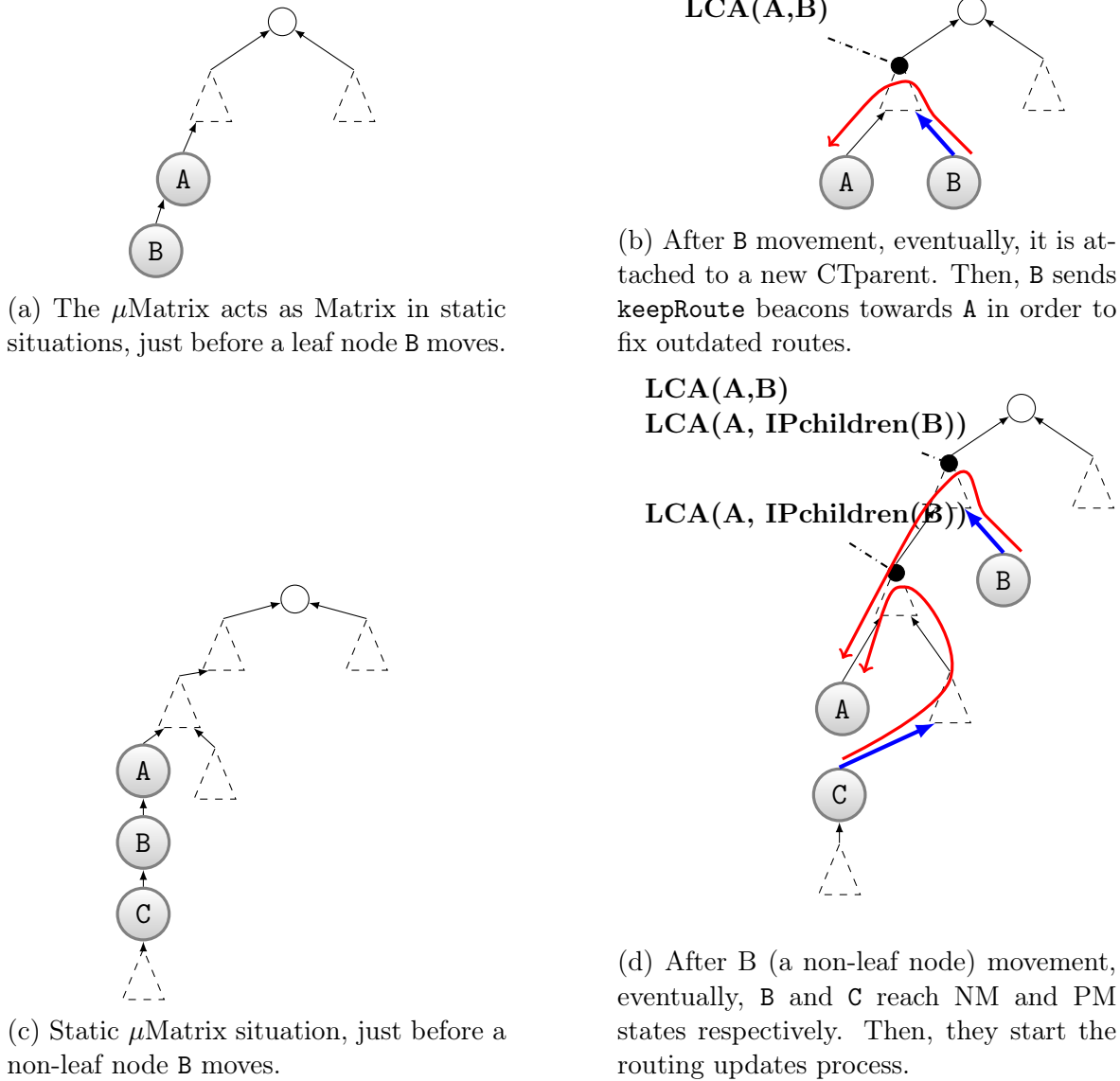


Figure 6.7: Mobile engine operation after mobility events.

detect a topology change, where $I_{min} \ll I_{max}$, the periodic and standardized Trickle approaches wait for at least $2 \times I_{max}$.

6.3.3.2 Mobile Engine: Finite State Machine

μ Matrix starts the mobile engine as soon as the host configuration finishes. The mobile engine allows the nodes to move around the 6LoWPAN. This module uses a finite state machine, as shown in Figure 6.6. Each node can be in one state depending on its previous condition and the knowledge about its neighborhood. The engine uses RevTT to recognize mobility and transit among states. Following, we discuss the actions taken

in each one of those states.

Each node starts at Home Location (HL) state. In HL, the nodes start RevTT sending a `hasMoved` beacon to its $CTparent(\eta)$. When RevTT identifies an inconsistency and triggers a mobility event, the node transits to Movement Detected (MD) state.

When a node reaches the MD state, it knows that a mobility event happened, but it does not know if itself or its parent moved. There are at least two ways to automatically find out who moved. First, a node can pro-actively query its children ($IPchildren(\eta)$), if no one answer then the node moved; otherwise the parent moved. Second, a node must wait for a period (e.g., one RevTT I_{max} interval) to receive `hasMoved` beacons from its children and then infer who moved. We use the second approach in our implementation. After discovering who moved, the node goes to Node Moved (NM) or Parent Moved (PM) state.

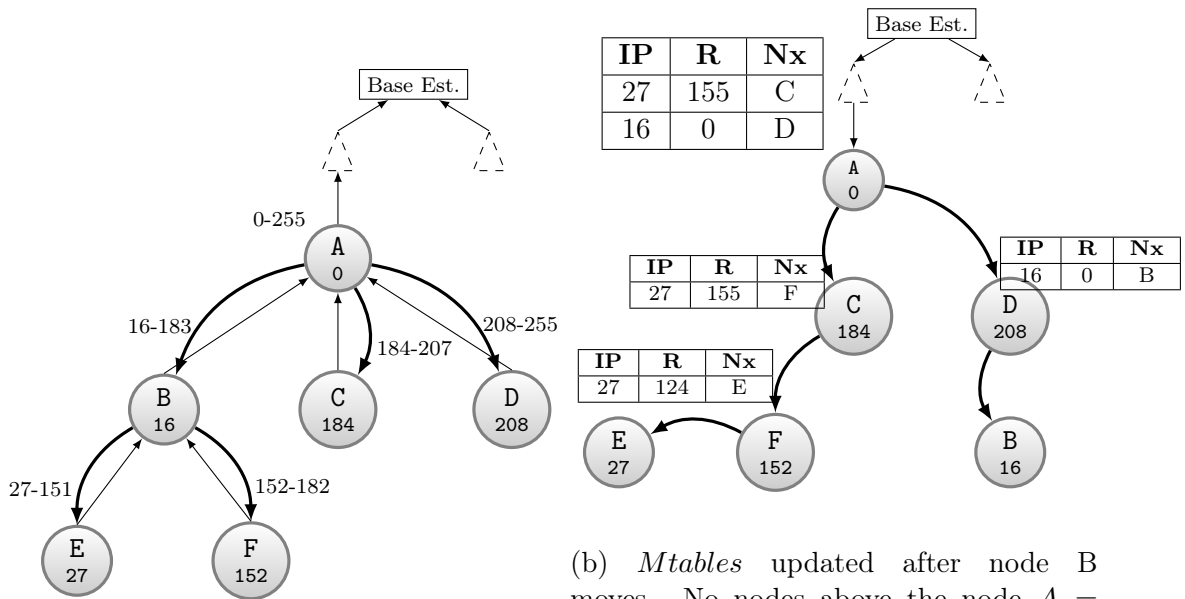


Figure 6.8: μ Matrix’s preserves locality when it updates the routing table under mobility events. *Mtables* above LCA do not need updates.

Several actions are taken when a node reaches NM state. Firstly, the node disables the $IPchildren(\eta)$ table usage due to the node new position in the Ctree. Next, the *Mtable* is cleaned, because it must be outdated. Then, the node triggers the new parent discovery from underlying collection protocol. When the node is at-

tached again to the Ctree, it restarts RevTT with new CTparent and begins sending `keepRoute.IP_ONLY` at a frequency of δ to its IPparent. At NM state, the nodes do not fill the `nodeInfo.IP_RANGE` field because the node is no longer at home location in the IPtree, being incapable of using its $IPChildren(\eta)$ table. Figures 6.7(a)(b) illustrate this situation. Figure 6.7a shows the node B before its movement, then when B is attached to a new CTparent (Figure 6.7b), it sends `keepRoute` beacons to A. The beacons are forwarded upward to the Lowest Common Ancestor $LCA(A, B)$ and then downwards to the node A.

When a node reaches PM, this means that its parent moved. Then, the node triggers the parent discovery mechanism from the underlying collection protocol. When it is attached again to Ctree, it restarts RevTT and starts sending `keepRoute` beacons (if it has children in the IPtree, it fills the `IP_RANGE`) at a frequency of δ . The first beacon is addressed to its grand IPparent. The Figure 6.7(c)(d) illustrates this situation. Figure 6.7c shows C, a non-leaf node, before B movement. Then, in Figure 6.7d, B moves and then C eventually reaches PM state, next C starts sending `keepRoute` beacons to its $grandIPparent = A$. The messages travel to $LCA(A, C)$ and then to the ultimate destinations. The node B moves to NM state and takes the actions accordingly.

Eventually, nodes return to their home position being attached again to its IPparent in the Ctree. This situation also triggers some actions. First, the node stops sending `keepRoute` beacons. Also, the returned node restarts the RevTT with its IPparent.

Besides the action in each state, optional actions can be made. If a node is attached to a sequence of CTparent before returning to the home location, several states will be installed in-network *Mtables*. Although the *Mtable*'s THL field exists to remove inconsistent entries, it is possible to send `rmRoute` beacons to each node's PRVparent to eliminate route inconsistency quickly.

Discussion: note that a sub-tree can move and nodes still hear each other. For instance, in Figure 6.7c suppose A and B move together. Then, A and C eventually will transit to PM, while B and C's sub-tree remain in HL. In this case, the LCA has two *Mtable* entries matching with C's sub-tree, but one more restrictive than other. Thus, the LCAs play a key role, which they always route through the most restrictive *Mtable* range match available.

Also, note that μ Matrix preserves locality when manages mobile nodes since no *Mtables* need updates above LCA. Figure 6.8 illustrates this situation. In Figure 6.8a, shows μ Matrix's routing structures in a static situation. Then, the node B moves, Figure 6.8b, it eventually reaches the NM state, therefore the *Mtables* comprised in the shortest path from B to $A = IPparent(B) = LCA(B, IPparent(B))$ will receive

B's `keepRoute` updates. Also, B movement causes E and F to transit from HL to PM state. Then, when E and F eventually find new routes, they will update the *Mtables* comprised between them and its `grandIPparent = A`.

Note that *Mtable(C)* and *Mtable(A)* require only one entry for both E and F sub-trees. We explore the fact of the contiguous IP ranges can be aggregated into unique entries in the *Mtables* improving the memory usage efficiency.

6.3.3.3 Loop Avoidance

Dynamic links and mobile nodes cause turn route information to become outdated, which may leverage routing loops [47]. μ Matrix uses data path validation and adaptive beaconing to detect loops such as CTP and RPL [47, 70]. Besides that, if a node receives more than one unique control packet⁵, then this indicates an inconsistency in the tree triggering the control packet suppression and the underlying collection protocol route update. Besides that, *Mtable* and `keepRoute` beacon have, respectively, THL and Time To Lived (TTL) fields, which are used to remove inconsistent routes and packets from the network.

6.3.4 Mobile Matrix Data Plane: Any-to-Any Routing

The Forwarding Engine (see Figure 6.2) is responsible for data traffic forwarding. Any-to-any routing combines both routing schemes: bottom-up and top-down. The engine uses bottom-up routes to forward packets until the LCA between the sender and receiver, and then it uses top-down routes to forward data to the destination. Upon receiving a data packet, the node checks if the message is for itself. Second, the node tries to match the destination with the most restrictive entry in the *Mtable*. Third, if any *Mtable* entry matches with the target address, then the node checks if the packet destination falls within some range in *IPchildren(η)*, if a match occurs, then the node forwards the packet downwards according. Finally, if all previous attempts fail, then the node sends the packet upwards using *CTparent(η)*.

6.4 Complexity Analysis

For the formal analysis, we assume a synchronous communication message-passing model with no faults. Thus, all nodes start executing the algorithm simultaneously and the time is divided into synchronous rounds, i.e., when a message is sent from

⁵The `keepRoute` fields together (see Section 6.3.2.1) denote a unique packet instance.

node v to its neighbor u in time-slot t , it must arrive at u before time-slot $t + 1$, and $d(v, u)$ is the shortest path length between v and u in $Ctree \cup IPtree \cup RCtree$. The performance of μ Matrix in faulty scenarios is analyzed through simulations in Section 6.6.

6.4.1 Memory Footprint

As described in Section 6.3, the temporary routing information needed to maintain mobility is stored in the *Mtable* data structure of some nodes. Each entry is kept for at most THL_{max} seconds, a time interval pre-configured by the network operator, and is deleted unless a `keepRoute` beacon is received. In the following theorem, we bound the total number of *Mtable* entries in the network, necessary to manage the routing of each mobile node $\mu \in CTree$.

Theorem 3. *The memory footprint to manage the mobility of one node $\mu \in Ctree$ with μ Matrix is $\mathcal{M}(\mu) = O(\text{depth}(Ctree))$.*

Proof. Consider a node $\mu \in Ctree$ that has moved from its home location in time-slot t_0 and returned in time-slot t_f . Consider the permanent $IPparent(\mu)$ and the temporary $CTparent_i(\mu)$ in time-slot $t_0 < t_i < t_f$. A routing entry for the temporary location of μ will be stored in the *Mtable* of every node comprising the shortest path between $IPparent(\mu)$ and $CTparent_i(\mu)$. Moreover, if μ has descendants in the IPtree, in other words, $k(\mu) = |IPchildren(\mu)| > 0$, then each node $c \in IPchildren(\mu)$ will send `keepRoute` beacons to their respective $CTparent_i(c)$, and a (temporary) routing entry will be stored in the *Mtable* of every node comprising the shortest path between $CTparent_i(c)$ and $IPparent(\mu)$. Therefore, the total memory footprint to manage the mobility of a node μ is:

$$\begin{aligned} \mathcal{M}(\mu) &= d(CTparent_i(\mu), IPparent(\mu)) + 1 \\ &+ \sum_{c \in IPchildren(\mu)} (d(CTparent_i(c), IPparent(\mu)) + 1) \\ &\leq (k(\mu) + 1) \times (\text{depth}(Ctree) + 1) \\ &= O(\text{depth}(Ctree)) \end{aligned}$$

□

Theorem 3 implies that the total memory footprint to manage the mobility of m nodes is $O(m \times \text{depth}(Ctree))$. Note that μ Matrix preserves locality when managing

mobile routing information of a node μ , since no *Mtable* needs to be updated at nodes above the $LCA(IPparent(\mu), CTparent(\mu))$.

6.4.2 Control Message Overhead

Control messages used by μ Matrix are comprised of three types: i) those used by Matrix to set up the initial IPtree and address allocation; ii) **keepRoute** beacons, defined in Section 6.3.2.1; and iii) **hasMoved** beacons, defined in Section 6.3.3.1.

For any network of size n with a spanning collection tree *Ctree* rooted at node r , the message and time complexity of Matrix protocol in the address allocation phase is $Msg(Matrix^{IP}(Ctree)) = O(n)$ and $\mathcal{T}(Matrix^{IP}(Ctree)) = O(depth(Ctree))$, respectively, which is asymptotically optimal, as proved in [95]. Next, we bound the number of control messages of type (ii) and (iii).

Theorem 4. *Consider a network with n nodes, with a spanning collection tree *Ctree* rooted at node r , and m mobility events, consisting of m nodes μ_i , changing location during time intervals $\Delta_i \leq \Delta$ time-slots. Moreover, consider the **hasMoved** beacon parameters I_{min} , I_{max} and I_k and the **keepRoute** beacon interval of δ time-slots. The control message complexity of μ Matrix to perform routing under mobility of m nodes is*

$$\begin{aligned} Msg(\mu Matrix(Ctree)) &= O\left(\frac{m \times I_k}{I_{min}} + \frac{n}{I_{max}}\right) \\ &+ O\left(\frac{m \times \Delta}{\delta} depth(Ctree)\right). \end{aligned}$$

Proof. Firstly, we bound the number of **hasMoved** (hM) beacons, which are sent periodically by all nodes in order to detect mobility events. As described in Section 6.3.3.1, when there is no mobility, the periodicity of **hasMoved** beacons is $1/I_{max}$. If some node μ has moved (an ack is lost), then I_k messages are sent at intervals of I_{min} time-slots. Using the fact that the network is a tree and the number of edges is $O(n)$, this gives a total of messages

$$Msg(\mu Matrix^{hM}(Ctree)) = O\left(\frac{m \times I_k}{I_{min}} + \frac{n}{I_{max}}\right).$$

Now, we bound the number of **keepRoute** (kR) beacons. As described in Section 6.3, mobile nodes send periodic **keepRoute** beacons at a frequency of δ to keep the *Mtables* up-to-date. Consider a node $\mu \in Ctree$ that has moved from its home location in time-slot t_0 and returned in time-slot t_f . Consider the $IPparent(\mu)$, $CTparent_i(\mu)$

in time-slot $t_0 < t_i < t_f$, and $\Delta = t_f - t_0$. When μ is attached to a $CTparent_i(\mu)$, μ sends `keepRoute` beacons at a rate of δ for at most Δ time-slots, such beacons travel through the shortest path $|(CTparent_i(\mu), IPparent(\mu))| \leq 2 \times depth(Ctree)$. Furthermore, if μ has descendants, i.e., $k(\mu) = |IPchildren(\mu)| > 0$, then each node $c \in IPchildren(\mu)$ will also send `keepRoute` beacons at a rate of δ for at most Δ time-slots, such beacons will travel the shortest path $|(CTparent_i(c), IPparent(\mu))| \leq 2 \times depth(Ctree)$. Therefore, the total control overhead to manage the mobility of a node μ is $\leq 2 \times depth(Ctree)(k(\mu) + 1)\Delta/\delta$, which results in

$$Msg(\mu Matrix^{KR}(Ctree)) = O\left(\frac{m \times \Delta}{\delta} depth(Ctree)\right).$$

Finally, the total control overhead is bounded by:

$$Msg(\mu Matrix) = Msg(\mu Matrix^{HM}) + Msg(\mu Matrix^{KR})$$

□

Once again $\mu Matrix$ preserves locality when managing mobile routing state of a node μ since no messages need to be sent to nodes above the $LCA(IPparent(\mu), CTparent(\mu))$.

6.4.3 Routing Path Distortion

We analyze the route length of messages, addressed to mobile nodes. Consider the underlying collection protocol (e.g., CTP or RPL), which dynamically optimizes the (bottom-up, or upwards) links in the collection tree $Ctree$, according to some metric, such as Four-bit wireless link estimation [40]. We define an *optimal route* length as the distance of the shortest path between (s, d) , comprised of the upwards links of the collection tree $Ctree$ and the downwards links of the union of the IPv6 address tree and the reverse-collection tree, i.e., $IPtree \cup RCtree$.

Theorem 5. *$\mu Matrix$ presents optimal path distortion under mobility, i.e., all messages are routed along shortest paths towards mobile destination nodes.*

Proof. Consider a mobile node $\mu \in Ctree$, which has moved from its home location in time-slot t_0 . Messages addressed to μ and originated by some node $\eta \in Ctree$ in time-slot $t_i > t_0$ can belong to traffic flows of three kinds: (1) bottom-up: $LCA_i(\mu, \eta) = \mu$; (2) top-down: $LCA_i(\mu, \eta) = \eta$; and (3) any-to-any: $LCA_i(\mu, \eta) \neq \mu \neq \eta$. In case (1), messages are forwarded using the underlying collection protocol, using the upwards

links of the collection tree $Ctree$, which is optimal. In case (2), messages are forwarded using $Mtables$ of η and its descendents, until reaching the mobile location of μ in some time-slot $t_f > t_0$. This path is comprised of the downwards links of $IPtree \cup RCtree$ in time-slot $t_0 < t_i \leq t_f$, which is the optimal route from η to the mobile location of μ in that time-slot. In case (3), the route between η and $LCA_i(\mu, \eta)$ falls into the case (1) and the route between $LCA_j(\mu, \eta)$ and μ falls into the case (2), for some $t_0 < t_i \leq t_j \leq t_f$, which is optimal. \square

6.5 Mobility Modelling

As previously discussed in Section 6.3.3, the protocol's mobility management was designed upon the assumption that devices hold a home location position, for which they eventually return after moving around in the cyber-physical space. Thus, μ Matrix presents better performance under mobility patterns that present this characteristic (see Section 6.4). Although μ Matrix can, in principle, also work without the home location assumption, it affects its memory efficiency. Moreover, other protocols, like MANET [24, 64, 97], are designed to deal with this feature accordingly.

Fortunately, the home location assumption is often present in mobility patterns ranging from human [53, 55, 81, 91] to non-human behavior [18, 84]. *Humans mobility pattern* tends to include group meeting dynamics and regularity. For example, people typically have a home, and they go to places nearby (meet friends, work, go shopping, etc.), then, eventually, they return to their homes. On the other hand, in a *non-human mobility pattern*, entities move and also can maintain an initial fixed position where they eventually come back. For instance, consider a team of autonomous robotic vacuum cleaners, assigned with the task of cleaning an office building. Usually, the robotic cleaners move randomly and, when the batteries are low, they return to their initial positions to recharge.

These characteristics fit well with the properties of μ Matrix, exploiting its performance gains over other solutions for mobile environments. However, there is a lack of available real mobility traces in such domains, usually due to privacy-related or technical issues. Opportunely, researchers have developed mobility models to fill in this gap [6, 53, 84]. A mobility model simulates the real mobility behavior and allows us to generate variable traces in several dimensions: spatial, temporal, and size. We employ mobility models to evaluate μ Matrix in different scenario conditions and highlight its potential to support IoT, IoMT, and SIoT. Following, we present the mobility model used in this work, its parameters, and behavior.

Table 6.2: GRM parameters.

Parameters	GRM-Inf06	GRM-Camb.	GRM-MIT
<i>Simulation parameters</i>			
# of nodes	78	54	100
Duration (days)	3	11	15
Group duration (h)	12	24	720
DIM (m ²)	300	500	1000
Path time	120	120	300
<i>Statistical parameters</i>			
α_{gmt}	1.35	1.35	2
β_{gmt}	12	24	720
α_{dur}	1.5	1.5	2
β_{dur}	3	20	720
α_{size}	2.24	2.24	2.24
β_{size}	30	30	30

6.5.1 Human Mobility Model

Hess et al. [53] survey available mobility models in the literature. Here, we highlight two: SWIM [81] and Group Regularity Mobility Model (GRM) [91]. SWIM produces synthetic traces with similar properties of real mobility traces. It assumes that humans go to places nearby their home, where they meet others, and eventually they return to their homes. GRM presents the same features of SWIM, but it introduces the dynamics of group meetings and social community structure. GRM produces synthetic traces with human and group regularity while other models do not.

These mobility models and others with similar characteristics [53, 84], especially the home location assumption, are suitable for μ Matrix. In this work, we use GRM as mobility model to generate traces based on real traces parameters. We produce three traces using GRM mobility model: GRM-Inf06, GRM-Camb., and GRM-MIT. Table 6.2 lists the GRM parameters for each trace⁶. The simulation parameters are self-explanatory, except by the path time which defines the time of a mobile entity takes to move from a location to another. The statistical parameters are parameters of truncated power laws with cut-off where α_* is the power law exponent and β_* the cut-off value: α_{gmt} and β_{gmt} define the group meeting times distribution parameters; α_{dur} and β_{dur} characterize the time that a group of entities will spend together; finally, α_{size} and β_{size} define which entities will be at each group meeting. The reader can find

⁶We extract the parameters from the following references:[19, 56, 81, 91].

more parameter details in [91]. Following, we describe the traces produced by GRM:

- *GRM-Inf06*: this trace was produced based on the *Infocom06* [56] real trace. The original trace was produced during a scientific conference. It has 78 nodes, of which 34 were assigned to 4 groups with sizes 4, 5, 10, and 5. The original conference trace has three different levels, but we simplify to 1 since GRM does not support this feature. The trace covers 3 days.
- *GRM-Camb*: we generate this trace based on the *Cambridge* [56] real trace, which uses 54 nodes (36 mobile + 18 fixed) distributed into two groups of students in the University of Cambridge. The data set covers 11 days.
- *GRM-MIT*: the GRM's authors provided a ready to go GRM-MIT trace [91]. The synthetic trace was produced based on *Reality MIT* trace [35] where 100 smart phones were deployed to students in two university buildings. This is the most representative trace in terms of a large area, and higher mobility. In the experiments (Section 6.6), we highlight the contrast between GRM-MIT and the other traces.

The above real traces and others can also be found in Crawdad site [66]. But, the mobility traces usually have only the contact trace (when two nodes meet each other) and the time when the contact happened. Most of the mobile network simulators use mobile traces of positions instead of contact traces. Thus, mobility models have the advantage of generating plausible locations (coordinates) of the nodes, instead of using heuristics to infer positions from contact trace [132].

6.5.2 Non-human Mobility Model

In the literature, there are several non-human mobility models available [6, 12, 84]. We highlight the RWP well-known mobility model to evaluate MANET routing protocols [12]. In RWP, the mobility entities move freely in a random direction, velocity, and acceleration. we propose the Cyclical Random Waypoint Model (CRWP), a mobility model based on the RWP. CRWP is useful to model scenarios where some of the entities move to different destinations, and eventually, they return to their initial positions, which is the case of objects (e.g., portable devices) that move in offices, universities, hospitals, factories, etc.

In CRWP, the entities move independently to random destinations and speeds as in RWP. When an entity arrives at the destination, it stops for a given time T_{pause} . A difference in CRWP is that after n chosen destinations, the mobile entity returns

Table 6.3: CRWP parameters.

Parameters	Values
Simulation time	1.5 h
# nodes	100 in grid
# traces	10 ^{traces} / _{scenario}
Dim (m ²)	400
Node speed	constant 4 m/s
CRWP T_{pause}	constant 300 s
CRWP $Stops$	Uni. Distribution (1,3)
CRWP Trace	Low Mod. High
PerMobNodes	5% 10% 15%

Table 6.4: CRWP mobility metrics.

Mobility Metrics (Avg)	CRWP-Low.	CRWP-Mod.	CRWP-High
# of link failures	1621	3057	4838
Link duration (s)	761.90	457.4	345
Node degree	4.12	4.36	4.44
Time for a link to fail (s)	227.6	216.1	204.5

to its initial position. Besides that, only $k\%$ of mobile entities are outside of their initial position in each instant of time. CRWP has four parameters: i) *PerMobNodes*: maximum percentage of entities that are mobile in each instant of time; ii) *Stops*: number of stops that the mobile entity do before returning to its original position; iii) *Speed*: speed which the mobile entity moves; iv) T_{pause} : the amount of time that the entity stays in a destination position.

We produce three different traces using CRWP by varying the parameter $k\%$ (percentage of mobile nodes away from home location): i) CRWP-Low; ii) CRW-Mod; iii) CRWP-High. Table 6.3 shows the parameters for each trace.

6.5.2.1 CRWP Mobility Scenario Analysis

We use the BonnMotion [6] to implement CRWP as well as to generate and analyze mobility traces. Table 6.3 presents the CRWP simulation parameters. We simulated scenarios where $n = 100$ mobile entities are assumed to be in an office or building, and they can move around and return to a predefined home position. The nodes are deployed in a grid. We divided the mobility scenarios into three groups: low, moderate, and high mobility. Table 6.4 presents the average of some mobility metrics [6] that characterize each scenario (low, moderate, and high mobility). We highlight that the

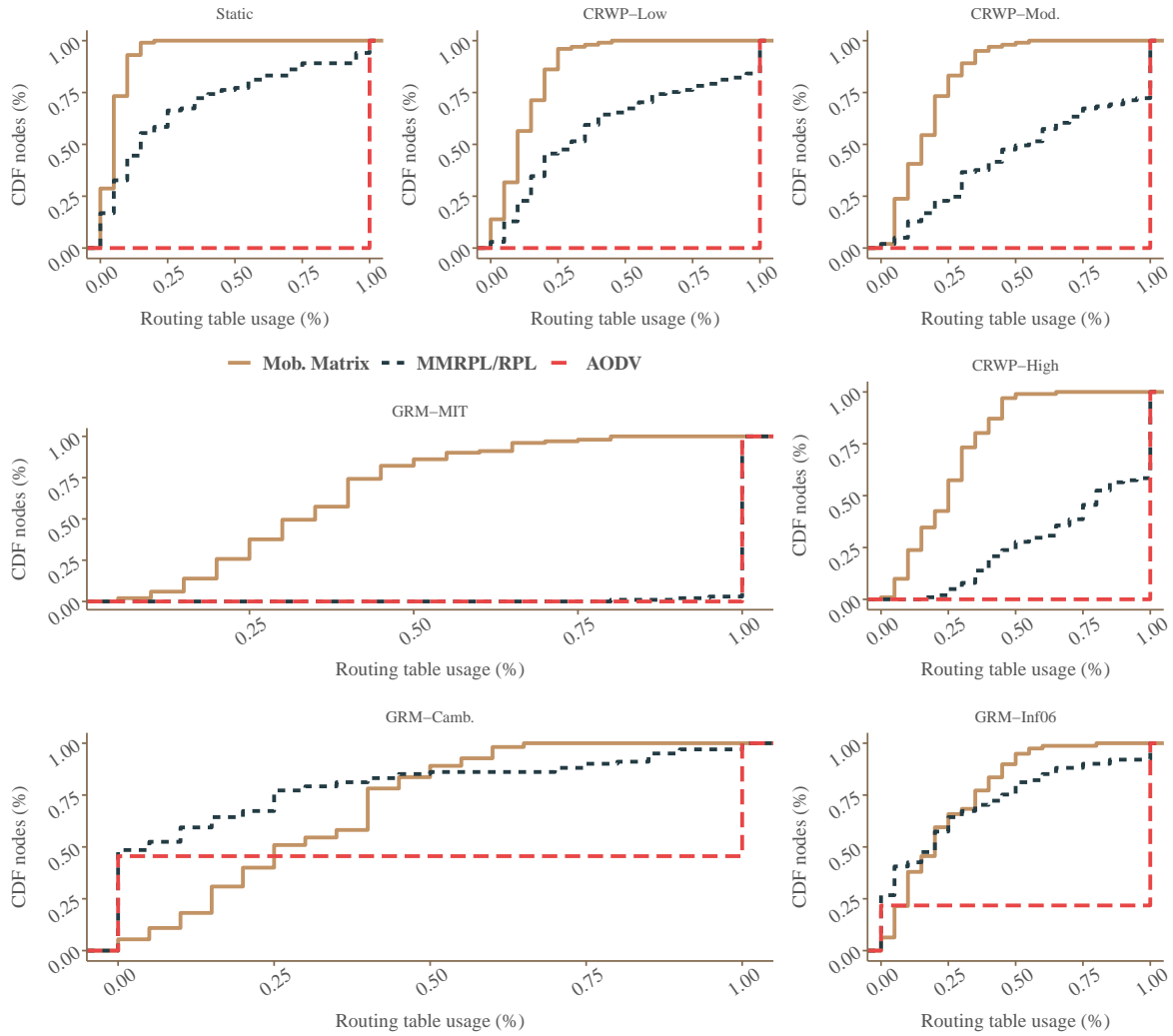


Figure 6.9: CDF of routing table usage. For μ Matrix *Mtable* + *IPchildren*, for RPL only downwards routing table. The maximum table size is 20.

metric number of link failures is an important metric in the performance of the network protocol. Observe that high mobility scenarios present up to 20% more topology changes than in low mobility. As expected, the average link duration decreases when *PerMobNode* increases. The metrics node degree and time for a link to fail do not vary much from each mobility scenario.

Table 6.5: Simulation parameters.

Simulation parameter	Values
Number of experiments	10 runs/trace
Border Router	1 center
keepRoute period	$\delta = 60$ s
RevTT	$I_{max} = 60$ s, $I_{min} = 1$ s, $I_k = 3$
RPL Trickle	$I_{max} = 60$ s
Downwards table	Size = 20 entries

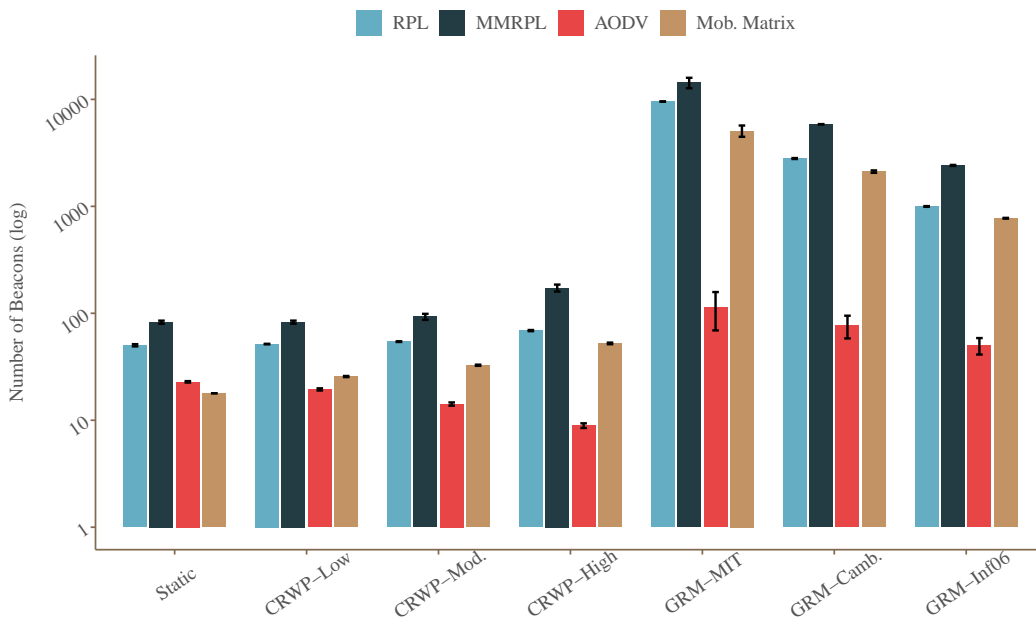


Figure 6.10: The number of control packets.

6.6 Evaluation

6.6.1 Simulation Setup

We implemented μ Matrix in ContikiOS [33] and made the source code publicly available⁷. The experiments were executed on Cooja [38] simulator and the following three baseline protocols were used: RPL, AODV, and MMRPL [25]. The former two protocols were already available on ContikiOS, while the latter we implemented on top of the available RPL implementation.

We simulated seven different scenarios. The first scenario represents the static network, in which devices do not move. One can interpret this scenario as the traditional static IoT. The remaining scenarios present different mobility patterns by using

⁷<https://bps90.github.io/matrix-code/>

CRWP and GRM as mobility models (see Section 6.5 for more details): 3 using CRWP named low, moderate, and high; 3 using GRM which we refer as Inf06, Camb., and MIT. One can interpret these scenarios as SIoT or IoMT cyber-physical mobile spaces being a step forward from standard IoT. Table 6.5 lists the default simulation and protocols parameters. In each plot, the bars or points represent the average, and the error bars indicate the confidence interval of 95%. The curves are the maximum table usage for a given mobility scenario.

For static and CRWP scenarios, we executed an application on top of the network layer, in which each node sends 20 data packets to the border router at a rate of 1 packet per minute. Upon receiving a data packet, the border router confirms to the sender with an acknowledgment packet that has the size of a data packet. The application waits for 10 min for protocols initialization and stabilization before it starts sending data. The nodes start sending their data in a simulation time randomly chosen in (10, 20] min. The mobility traces were configured to start after the stabilization time. Additionally, we generate 10 mobility traces for each scenario. Each trace and the static scenario were run 10 times, totaling 3010 executions.

We ran a similar application for GRM scenarios except that the nodes, after sending their data packets, reschedule to send the data packets again in a simulation time randomly chosen in next [1, 12] hours. This process repeats indefinitely until the maximum GRM simulation time is achieved. For each trace, we run 5 times, totaling 15 executions.

6.6.2 Results

In Figure 6.9, we show the Cumulative Distribution Functions (CDFs) of the percentage of downward routing table usage among nodes, for each scenario. In the static scenario, all μ Matrix nodes use up to 25% of available downwards route entries, while in RPL and MMRPL $\geq 75\%$ of nodes use $\geq 25\%$ of entries. MMRPL and RPL present almost identical routing memory usage since the only different between MMRPL RPL is the mobility detection mechanism, which does not affect memory usage. In AODV, the devices flood the network with route queries to find the packet's destination, then the devices opportunistically fill all available route entries. Therefore, AODV $\approx 100\%$ of nodes use 100% of route entries. Indeed, for some RPL nodes and almost AODV nodes, 100% of table entries are used. Usually, these nodes that use more memory are near the border router, and they play a fundamental role in top-down routing. If they overflow their downward routing table, then the traffic pattern top-down suffers from poor reliability, and some nodes may be unreachable.

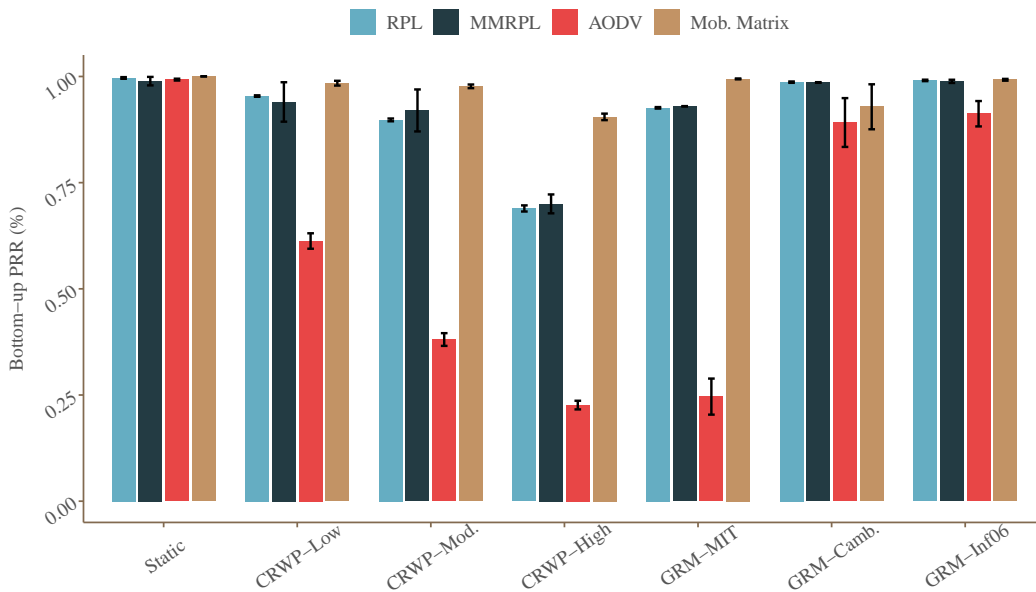


Figure 6.11: Bottom-up routing success rate.

μ Matrix also presents a more efficient memory footprint than other protocols, in non-human mobility scenarios (CRWP). The difference among the protocols grows up as the nodes mobility increase. Figure 6.9 shows CRWP- $\{$ Low, Mod., High $\}$ memory usage. In these scenarios, μ Matrix uses up to 65% of the downward routing table, while $> 15\%$ of RPL devices present full table usage as well as almost all devices running AODV.

For human mobility scenarios (GRM), we highlight two of them: GRM-MIT and GRM-Camb.. The first one, it presents higher mobility, larger area, number of nodes and duration than other scenarios. On the other hand, GRM-Camb. presents fewer nodes and mobility than MIT. Figure 6.9 also shows the protocols under the GRM mobility pattern. In GRM-MIT, μ Matrix presents lower downward routing table than RPL and AODV. For RPL and AODV, almost all route entries are used causing poor reliability in top-down and any-to-any routing. Therefore, μ Matrix is more efficient in the most dynamic scenario. In the GRM-Camb. scenario, RPL and AODV seem to presented better results, but GRM-Camb. has only 54 nodes and the routing table size is only 20, and yet some nodes consume all routing memory available. In GRM-Camb., μ Matrix uses $\leq 65\%$ of downwards route entries.

Another relevant analysis is the control messages (or beacons) overhead of each routing protocol. μ Matrix sends beacons to react to topology changes quickly by using the RevTT algorithm and the underlying collection protocol beaconing scheme. The protocol AODV sends route queries to find routes between the sender and receiver.

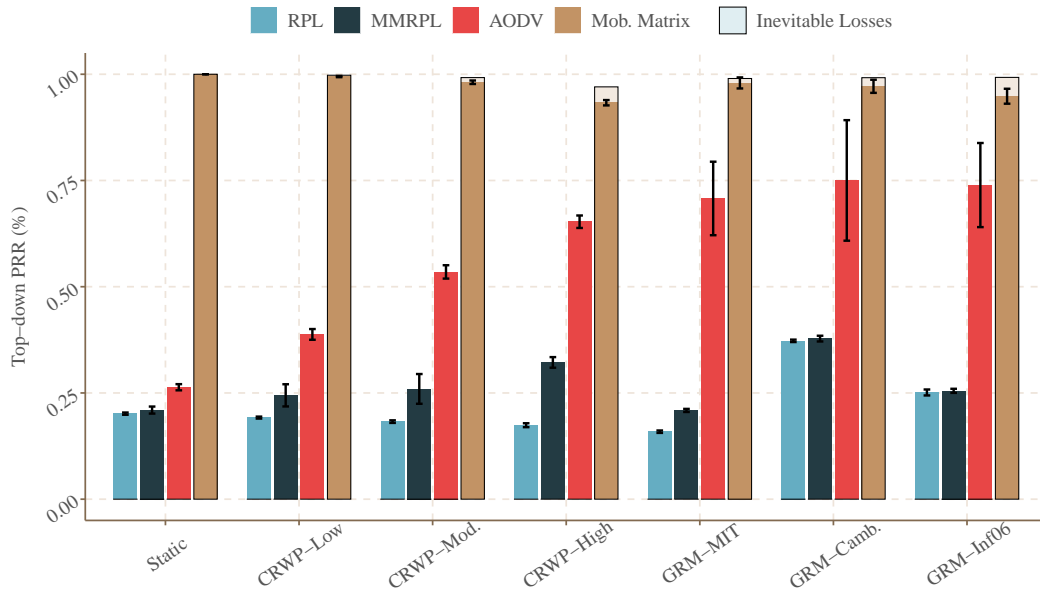


Figure 6.12: Top-down routing success rate. The transparent bar represents inevitable losses.

RPL sends control beacons to build and maintain its routing structures. μ Matrix allows tuning the RevTT fire rate to reduce the sending beacons, but note that the reverse trickle adjustment faces a trade-off between quick mobility discovery and control overhead. In Table 6.5, we set I_{max} of RPL and μ Matrix evenly and close to data packet rate, which gives to the protocols the fair opportunity to identify topology changes and react to them.

Figure 6.10 shows the amount of control traffic overhead of the protocols (the total number of beacons sent during the entire simulation). AODV is a reactive routing protocol (create routes on demand); therefore it sends fewer control packets than μ Matrix, MMRPL, and RPL, which are pro-active. However, AODV presents higher losses than others evaluated protocols as we show ahead. MMRPL, μ Matrix, and RPL present close control overhead being μ Matrix slightly more economical. The difference between RPL and μ Matrix does not exceed 8.6%. The difference of quantities in GRM traces to others is only due to the simulation time.

Figure 6.11 shows the Packet Reception Rate (PRR) in bottom-up data traffic. In all scenarios, μ Matrix presents higher or equal PRR than RPL, MMRPL or AODV. Upon node mobility, μ Matrix realizes that a topological change happened by using RevTT, it quickly triggers the underlying route discovery, and as a consequence, bottom-up routes are rapidly rebuilt, and the reliability increases. MMRPL and RPL also present high reliability on bottom-up data traffic overall evaluated scenarios but be-

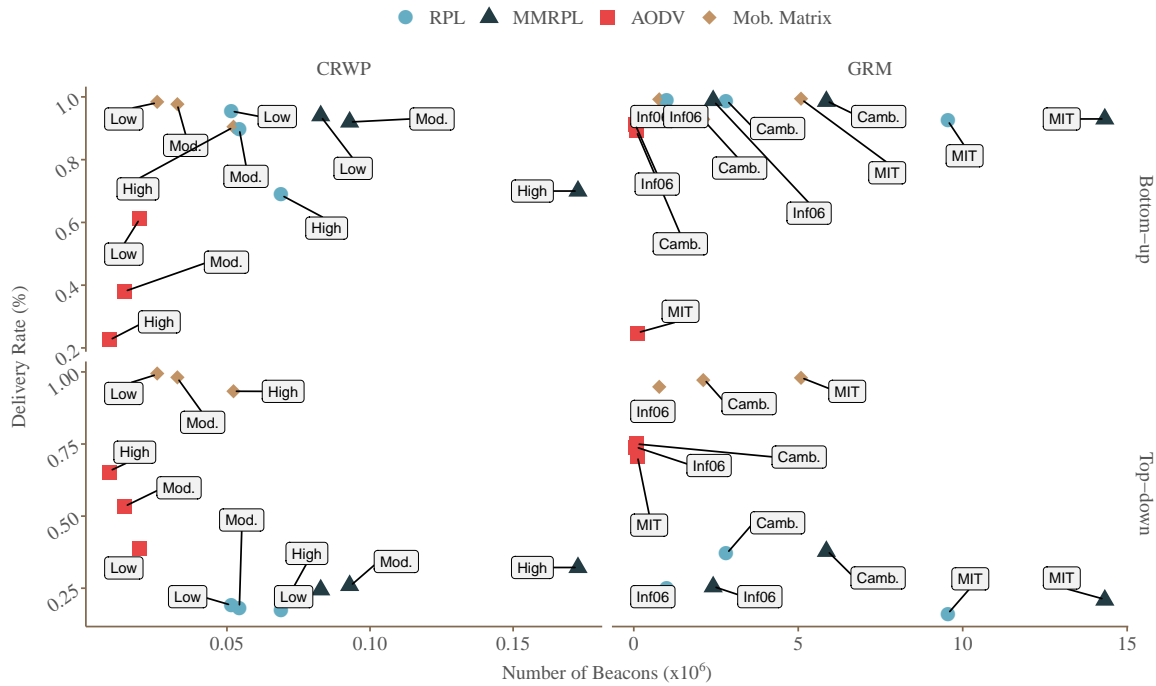


Figure 6.13: The trade-off between control message overhead and the successful delivery rate.

ing slightly less reliable than μ Matrix. AODV, in the static scenario, presents $\approx 100\%$ PRR, however, in non-human mobile scenarios, its reliability decreases as the mobility increases. In human mobility, for example, GRM-MIT (higher mobile scenario), AODV also presents poor reliability in bottom-up routing.

Figure 6.12 shows the PRR for top-down data traffic. In the plot, “inevitable losses” are represented with a transparent bar, and they refer to the number of messages that were lost due to the node being in transit from one location to another. Meanwhile, messages were routed before the route updated mechanism (see Section 6.3), in which case, there was no valid path to the destination, and the packet loss is inevitable.

It is possible to see that in all scenarios μ Matrix presents higher PRR than other protocols in top-down data traffic routing. Under no mobility, μ Matrix presented 99.9% of success rate, while RPL and MMRPL presented $< 21\%$, and AODV presented $\approx 26\%$. In non-human mobility scenarios, μ Matrix PRR decreases slowly when more mobility is allowed. In the harshest mobility scenario, CRWP-High, μ Matrix shows PRR of 95% while AODV has 68% and RPL has 17%. RPL, MMRPL, and AODV presented top-down PRR 19%, 28%, and 68% respectively. μ Matrix, in GRM scenarios, presented at least 97% of top-down PRR, and RPL exhibited the lowest PRRs ranging from 16% to 35% followed closely by MMRPL. AODV presented PRR up to 75%, but its delivery rate with acknowledgment is low as we show ahead.

RPL, MMRPL and AODV suffer from poor reliability because of the lack of memory (see Figure 6.9) to store top-down routes, while μ Matrix is more efficient in the memory usage, as we have shown in complexity analysis in Section 6.4.

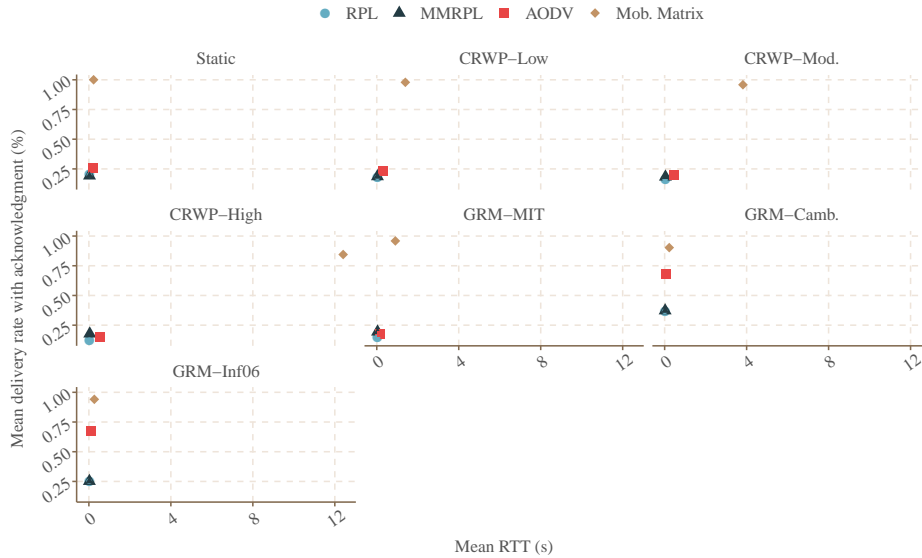


Figure 6.14: The trade-off between the delivery rate with acknowledgment and RTT.

Figure 6.13 shows the trade-off between control message overhead and the successful delivery rate. Figure 6.13 is composed of four graphics. The two top graphics are for bottom-up traffic, and the two lower graphics are for top-down traffic. The two left graphics are for CRWP mobility model, and the two right graphics are for GRM mobility model. In each graphic, it is desirable a high delivery rate with a low number of beacons (upper-left region). However, to identify mobility passively and quickly, usually, it requires more control messages. Therefore, protocols that balance this trade-off are fundamental in the IoT, IoMT or SIoT context, especially when the devices have energy constraints.

Note that in all scenarios and traffic patterns, μ Matrix presents a higher delivery rate and a balanced number of beacons. AODV is the most economical concerning message overhead, but it suffers from reliability, especially in high mobility scenarios and both data traffic pattern. RPL shows the lower delivery rate and higher control message overhead. Also, the MMRPL and RPL beacons go from the node to the border router to recreate routes downwards. μ Matrix reduces this cost by preserving route updates locality as discussed in Section 6.3 and analyzed in Section 6.4. In general, MMRPL presented a better delivery rate than RPL at the cost of more beacons.

Figure 6.14 depicts the trade-off between delivery with acknowledgment rate and Round Trip Time (RTT). Note that only round-trip messages were considered. We plot

a graphic for each of the seven mobility scenarios. RPL, MMRPL, and AODV suffer from losses in the top-down traffic (as explained in Figure 6.12). μ Matrix presents the higher delivery with acknowledgment rate. The mean RTT is similar between the four protocols except for CRWP-High scenario. In the CRWP-High scenario, μ Matrix presents higher RTT because, when there is no valid route between the sender and the receiver, CTP [47] (underlying routing protocol employed) keeps some messages in a buffer for a while, then the forwarding engine (see Section 6.3) eventually sends the messages. Therefore, in high mobile scenarios where the topology constantly changes, some messages will be delivered with some delay; thus μ Matrix also presents higher delivery with acknowledgment rate.

6.7 Concluding Remarks

In this work, we have designed, analyzed and evaluated the Mobile Matrix, a mobile routing protocol with a hierarchical addressing scheme for resource-constrained devices largely employed in IoT, IoMT, and SIoT. In the new IoT context, the “things” are able to move and do social ties; thus μ Matrix represents a step towards this new mobile cyber-physical environment by allowing the devices to move around while providing device mobility transparency to upper layers in the network stack. The protocol has a low memory footprint, adjustable control message overhead, optimal routing path distortion, and provides any-to-any communication. We provide a formal analysis of μ Matrix memory footprint, control message overhead, and the routing path distortion. We also introduced the CRWP, a non-human mobility model suited for scenarios with mobile devices that have cyclical movement patterns.

We evaluated the routing protocols under human and non-human mobility patterns. Our μ Matrix implementation offers $\geq 95\%$ of top-down PRR in highly dynamic and mobile scenarios, while other protocols $\leq 75\%$. This difference is a consequence of the downwards routing table usage, in which the devices running μ Matrix protocol use up to 65% of routing entries available while for RPL, MMRPL and AODV several devices presented full routing table implying in poor top-down and any-to-any reliability. It was also shown that existing routing protocols have poor delivery with acknowledgment rate.

We have shown that existing routing protocols do not meet several IoMT and SIoT requirements such as mobility management, memory, and energy efficiency in routing. Thus, efforts in that direction, e.g., the μ Matrix, enable several opportunities for future research in IoMT and SIoT network stack support. For instance, it would be

interesting to evaluate: i) how μ Matrix and others ready-to-go protocols perform under IoMT or SIoT applications in a large-scale network with thousands of mobile devices; ii) how to passively detect mobility, the efficacy of this technique may lead to a better energy efficiency of routing protocols for IoMT and SIoT. It is also worth mentioning the possibility of extending μ Matrix to allow devices to move between different network domains.

Chapter 7

Conclusions and Future work

In this chapter, we summarize the main results achieved so far (Section 7.1) and present the open problems and future work derived from this thesis (Section 7.3).

7.1 Conclusions

In this thesis, we have studied distinct aspects related to routing under mobility in Internet of Things. Specially, we did studies, analyses, and formulated proposition of solution in Mobility detection, Handover process, and Mobility Management in IoT. Such studies were presented along this thesis, which are summarized in the Sections 7.1.1 and 7.1.2.

7.1.1 Mobility Detection

We have studied the mobility detection aspect in IoT from the lens of beaconing timer schemes at the routing layer level (See Chapter 4). Such a process is a key step of the design and overall performance of routing under static and dynamic IoT environments. An efficient timer scheme enables the routing protocol to be aware timely to network dynamic links while should be saver regarding constrained resource such as energy or channel occupancy. Thus, the timer scheme must deal fairly with this basic trade-off. If it is too greedy by sending advertisement frequently, it responds quickly to topology changes, but it spends energy and introduces an overhead to the wireless shared channel. However, if the timer scheme is too slow, i.e., it sends advertisement infrequently, it will save energy and bandwidth, but topology problems will persist for a long time.

Commonly, routing protocols rely on only one timer schemes and assumes that all networked devices follow the same mobility behavior. We have argued that such assumptions do not fit. In this sense, we have proposed the Dribble learn-based timer scheme selector. Dribble uses learning model to predict devices mobility pattern and then it sets, to the device, a custom timer schemes (Periodic, TT, and RevTT) that better catches its mobility pattern. In our experiments (Section 4.5), we showed that Dribble is an alternative to better balance the trade-off faced by single timer schemes and it favors the delivery data rate while being aware of the device's power constraints.

7.1.2 Handover Process and Mobility Management

We have proposed two routing protocols for IoT. Initially, we proposed the Matrix routing protocol and later Mobile Matrix routing protocol. They share the same working core, however, they differ regarding their main goal. In short, both protocols were built upon the idea of using hierarchical IPv6 address allocation that explores cycle-free network structures. With this idea, Matrix and Mobile Matrix are able to perform any-to-any routing (especially top-down routing see Sections 5.3 and 6.3) by using only one-hop information in static conditions which means be aware of IoT device's resource constraints (especially, memory and energy). Matrix was designed to working properly in networks with low link dynamics, although Matrix implements mechanisms of reliability and fault-tolerance. Mobile Matrix does a step further being able to manage mobile nodes. It does that by only assuming that nodes have a home location for which the nodes eventually come back.

We bounded the memory footprint, control overhead, behavior upon network failures, and routing path distortion for our routing protocols proposition. We did this through mathematical analysis and experimental evaluations. Also, we made comparisons of Matrix and Mobile Matrix against the latest routing protocols, including RPL the *de facto* standard routing protocol for IoT in Low-Power and Lossy Networks. Our experiments (see Sections 5.5 and 6.6) have shown that our proposition can be an alternative for routing for IoT in static and mobile scenarios. The results indicated that Matrix and Mobile Matrix are favorable in terms of memory footprint, data delivery rate, fault-tolerance, and energy.

7.2 Publications

So far, during my Ph.D., I have authored or participated in the scientific publications below:

Chapter 1 and 2: only part.

1. Santos, B. P., Silva, L., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. (2016). Internet das coisas: da teoria à prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. **Book Chapter Accepted.**

Chapter 4:

2. Santos, B. P., Rettore, P. H. L., Vieira, L. F. M., and Loureiro, A. A. (2018c). Dribble: a learn-based timer scheme selector for mobility management in iot. *IEEE Wireless Communications and Networking Conference WCNC*. **Conference paper.**

Chapter 5:

3. Peres, B. S., Souza, O. A. d. O., Santos, B. P., Junior, E. R. A., Goussevskaia, O., Vieira, M. A. M., Vieira, L. F. M., and Loureiro, A. A. F. (2016). Matrix: Multihop Address Allocation and Dynamic Any-to-Any Routing for 6LoWPAN. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '16*, pages 302–309, New York, NY, USA. ACM. **Conference paper.**
4. Peres, B., Santos, B. P., de O. Souza, O. A., Goussevskaia, O., Vieira, M. A. M., Vieira, L. F. M., and Loureiro, A. A. F. (2018). Matrix: Multihop Address allocation and dynamic any-To-any Routing for 6LoWPAN. *Computer Networks*, 140:28 – 40. **Published Journal.**

Chapter 6:

5. Santos, B. P., Goussevskaia, O., Vieira, L. F. M., Vieira, M. A. M., and Loureiro, A. A. (2017a). Mobile Matrix: A Multihop Address Allocation and Any-to-Any Routing in Mobile 6LoWPAN. In *Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet '17*, pages 65–72. **Conference paper.**
6. Santos, B. P., Goussevskaia, O., Vieira, L. F., Vieira, M. A., and Loureiro, A. A. (2018a). Mobile Matrix: Routing under Mobility in IoT, IoMT, and Social IoT. *Ad Hoc Networks*. **Published Journal.**

Others scientific publication beyond this thesis scope: in Intelligent Transportation Systems, Social Networks, and IoT/WSN:

7. Rettore, P. H., Santos, B. P., Campolina, A. B., Villas, L. A., and Loureiro, A. A. (2016). Towards intra-vehicular sensor data fusion. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 126–131. IEEE. **Conference paper.**
8. Cunha, F., Maia, G., Ramos, H., Santos, B. P., Celes, C., Rettore, P., Campolina, A., Guidoni, D., Souza, F., Villas, L., Mini, R., and Loureiro., A. (2017a). *Emerging Trends in Vehicular Ad-hoc Network (VANET)*. Emerging Wireless Communication and Network Technologies: Principle, Paradigm and Performance. Springer Chapter Book. **Book Chapter Accepted.**
9. Cunha, F., Maia, G., Ramos, H., Santos, B. P., Celes, C., Rettore, P., Campolina, A., Guidoni, D., Souza, F., Villas, L., Mini, R., and Loureiro., A. (2017b). *Sistemas de Transporte Inteligentes: Conceitos, Aplicações e Desafios (Intelligent Transportation Systems: Concepts, Applications and Challenges - In portuguese)*. XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Ied.Porto Alegre: Sociedade Brasileira de Computação (SBC) SBRC 2017. Chapter Book. **Book Chapter Accepted.**
10. Santos, B. P., Vieira, L. F. M., and Vieira, M. A. M. (2017c). CGR: Centrality-based green routing for Low-power and Lossy Networks. *Computer Networks*, 129:117–128. **Published Journal.**
11. Santos, B. P., Rettore, P. H., Ramos, H. S., Vieira, L. F., and Loureiro, A. A. (2017b). T-MAPS: Modelo de Descrição do Cenário de Trânsito Baseado no Twitter. *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. **Conference paper.**
12. Santos, B. P., Rettore, P. H., Ramos, H. S., Vieira, L. F., and Loureiro, A. A. (2018b). Enriching Traffic Information with a SpatiotemporalModel based on Social Media. In *Computers and Communications (ISCC), 2018 IEEE Symposium on*, page 0. IEEE. **Conference paper.**

Also, we have also submitted a full journal.

13. Rettore, P. H., Santos, B. P., Maia, G., Villas, L. A., and Loureiro, A. A. F. (2018). Rode: Road data enrichment framework based on heterogeneous data fusion for

its. *Transactions on Intelligent Transportation Systems*. **Journal under minor review.**

Finally, we are working to submitting new results obtained from Chapter 4 in a full journal.

7.3 Open Problems and Future Work

During this research, we perceived many future directions on this thesis. We highlight some of them below.

7.3.1 Inter-domain Routing with Mobile Matrix

Initially, μ Matrix was designed to provide any-to-any intra-domain routing for static and mobile nodes. However, the mobile freedom degree of a mobile IoT allows mobile devices to move through different domains. On the Internet, inter-domain mobility was handled by deploying centralized mobile anchors for which inter-domain mobile nodes eventually attach. However, with the growth of the Internet and the increasing number of IoT devices (potentially billions of nodes) attached to the network, it is worth that distributed models for mobility management will be needed.

On the Internet, the IPs are used to the routing process as well as a mechanism to identify end hosts. One way to implement a mobile management inter-domain is through locator-identifier separation. Locators are the address used to identify border routers in the network. These addresses are advertised to the network (e.g., by using BGP). Identifiers are addresses to identify end hosts, and they are not spread out to the routing system.

One example of locator-identifier implementation is Locator/Identifier Separation Protocol (LISP) [39]. Figure 7.1 shows the LISP packet flow. Host at the same network domain can exchange packets directly ($A \rightarrow B$), border routers (ITRx and ETRx) uses the routing system (e.g., BGP) to advertise its address (locator address) to the global Internet. Note that end hosts do not advertise its identifiers. If A wants to send packets to C, then it first sends the packets to the border router (ITR2), in turn, ITR2 queries the LISP mapping system to retrieve the location border routers that can reach C (e.g., ETRx). Then ITR2 encapsulates the A's packets and forwards them to the retrieved border router locator (e.g., ETR1). Then, ETR1 removes the headers of the encapsulated packet and forwards the regular A's packets towards C. Note that LISP uses a centralized approach for the mapping location-identifier, but

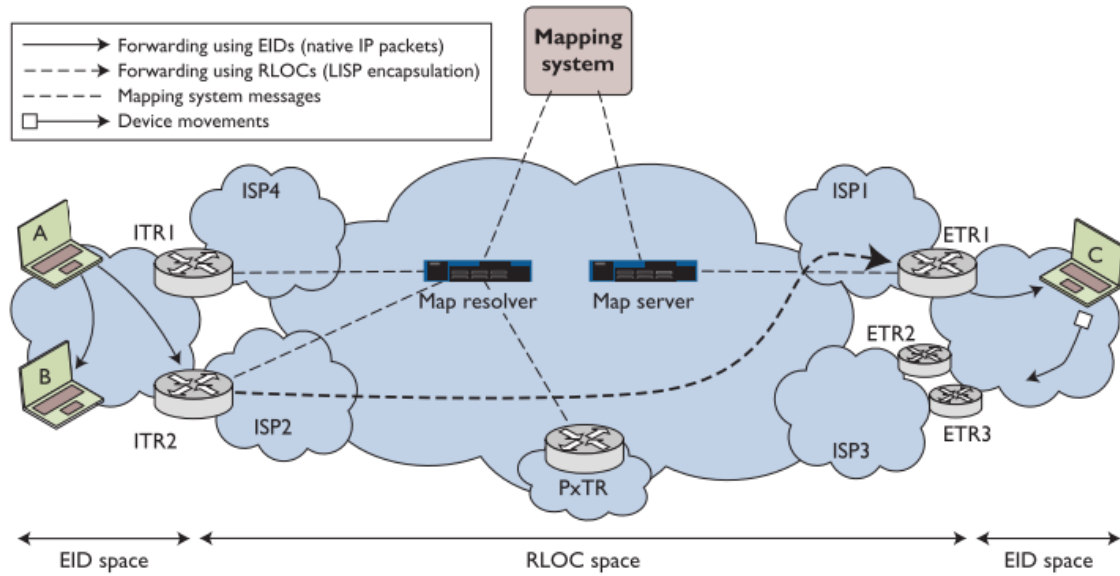


Figure 7.1: Lisp packet flow. (EID: endpoint identifier; ETR: egress tunnel router; ITR: ingress tunnel router; PxTR proxy tunnel router; RLOC: routing locator). Source: [115].

there is a distributed approach namely Distributed Mobility Management scheme in Locator/Identifier Separation networks (DMMLIS) [100].

We are interested in investigating the viability of extension of μ Matrix to couple inter-domain routing feature, especially in a distributed fashion. One way to tackle this issue is by performing an extensive literature study, as well as studying the benefits and issues of distributed mobile inter-domain management as listed in [20]. If viable, we want to perform a formal analysis, and comparative evaluation of an inter-domain enable μ Matrix routing with state-of-the-art solutions.

7.3.2 Social IoT: A Mobility Model

As mentioned in Chapter 2, SIoT is a new IoT paradigm which expands and provide several benefits. However, we argue that we are not prepared yet to support the paradigm fully. One argument is that the way smart devices move play a key role in the understatement and design of solutions for SIoT. However, real smart device mobility traces are not yet available. Thus, synthetic approaches is a way to overcome this limitation [9]. In this sense, one can analyze and eventually propose a mobility model to entities in SIoT to generate social traces that can benefit SIoT application and support software designs.

7.3.3 Mobile Agents and IoT

Mobile Agent (MA) has been a subject of study since 2000 when the Internet became more dynamic, mobile and ubiquitous with the introduction of embedded systems [80]. There is not a consensus about a definition of MA, but in general lines, *Mobile Agent* is a software that can move within a heterogeneous network (Internet, intranet) and by receiving authorization levels, it acts autonomously on behalf of or in place for some entity. MA has been used to a range of purposes such as find and filter information, automate work, cooperative tasks, automatic processes without user intervention [15, 80]. MA has a life-cycle model (creation/destroy, start/stop, etc.), a computational model (data manipulation, thread control), a security model (the way that MA access network resources), and a communication model (communication between agents and between agent and others entities), navigation model (cares about the transportation of the agents between computational resources in different physical locations).

Although MA has been studied on a diversity of contexts, on the IoT domain, MA has received little attention. However, we believe that MAs have certain benefits that are especially useful for IoT context, for example, MA help to reduce network load, network latency, and encapsulate protocols. Some literature studies have shown that MA concept can be brought to the IoT domain. In [71], the authors propose MA to integrate IoT and WSN, and in [127], it is described the technical solutions used to implement MA for Web-of-Things by using HTML5. However, MA studies on IoT context are still in embryonic stage.

Aiming do a step forward, we draw some research questions that we believe to be worth as future work direction.

- *How to potentialize device mobility by using MAs (software mobility)?*

Mobile agents have a navigation model [15, 99] which defines how the mobile agents will move, e.i., the itinerary of the MA. One can study the viability of potentializing the hardware mobility by taking advantage of MA movements.

Our previous experiences with centrality metrics and routing in WSN/IoT [112] allow us to rise up one specific question involving centrality and navigation model that is: *is it possible to use centrality metrics to plan itineraries for MA in IoT context?*

- *What is the optimal number of MA in an IoT system?*

It is a fact that multiple agents can co-exist in a system. However, it is a challenge to determine how many agents (or clones) can co-exist. The life-cycle, commu-

nication, and navigation models play a key role in the determination of such an optimal number.

7.3.4 IoT on 5G context

Another viable direction is the symbiosis between IoT and 5G¹. The 2G of a mobile network was designed to voice, 3G for data exchange, and 4G for widely Internet access. Now, 5G aims to address the limitations of the previous generation such as energy consumption, machine-to-machine communication, and speed (over 10× faster than 4G). Also, 5G has been designed for mobile low-power and constrained devices as well. Thus, these 5G features have the potential key to enabler IoT. IoT powered by 5G will use millimeter waves (for high speed communication and accommodate billions even trillions of devices), small cells (for connectivity), MIMO, beamforming (for reduction of interference and improvements on efficiency), and full-duplex (more efficiency on communication).

The symbiosis of IoT and 5G rise up several issues that request for solutions, especially regarding mobility management. In [4], the authors highlight that a critical research challenge in IoT/5G networks is the mobility management, coverage, and reachability which remain an open research area for IoT. Besides that other issues are the scalability, network management, interoperability under heterogeneous devices, security and privacy, and network congestion and overload.

¹5th generation wireless systems.

Bibliography

- [1] Afzal, B., Umair, M., Shah, G. A., and Ahmed, E. (2017). Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges. *Future Generation Computer Systems*.
- [2] Agnihotri, S. and Ramkumar, K. (2017). A survey and comparative analysis of the various routing protocols of Internet of things. *International Journal of Pervasive Computing and Communications*, 13(3):264–281.
- [3] Akpakwu, G. A., Silva, B. J., Hancke, G. P., and Abu-Mahfouz, A. M. (2017). A survey on 5G networks for the Internet of Things: Communication technologies and challenges. *IEEE Access*, 6:3619–3647.
- [4] Akpakwu, G. A., Silva, B. J., Hancke, G. P., and Abu-Mahfouz, A. M. (2018). A survey on 5g networks for the internet of things: Communication technologies and challenges. *IEEE Access*, 6:3619–3647.
- [5] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376.
- [6] Aschenbruck, N., Ernst, R., Gerhards-Padilla, E., and Schwamborn, M. (2010). BonnMotion: a mobility scenario generation and analysis tool. In *EAI ICST*, page 51.
- [7] Ashton, K. (2009). That ‘internet of things’ thing. *RFiD Journal*, 22(7):97–114.
- [8] Asl, H. Z., Iera, A., Atzori, L., and Morabito, G. (2013). How often social objects meet each other? Analysis of the properties of a social network of IoT devices based on real data. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 2804–2809. IEEE.
- [9] Atzori, L., Carboni, D., and Iera, A. (2014). Smart things in the social loop: Paradigms, technologies, and potentials. *Ad Hoc Networks*, 18:121–132.

- [10] Atzori, L., Iera, A., Morabito, G., and Nitti, M. (2012). The social internet of things (siot)—when social networks meet the internet of things: Concept, architecture and network characterization. *Computer networks*, 56(16):3594–3608.
- [11] Baccour, N., Koubâa, A., Mottola, L., Zúñiga, M. A., Youssef, H., Boano, C. A., and Alves, M. (2012). Radio Link Quality Estimation in Wireless Sensor Networks: A Survey. *ACM Transactions on Sensor Networks (TOSN)*, 8(4):34:1–34:33.
- [12] Bai, F. and Helmy, A. (2004). A survey of mobility models. *Wireless Adhoc Networks*.
- [13] Bellier, L., Malki, K. E., Castelluccia, C., and Soliman, H. (2008). Hierarchical Mobile IPv6 (HMIPv6) Mobility Management. RFC 5380.
- [14] Bezunartea, M., Gamallo, M., Tiberghien, J., and Steenhaut, K. (2016). How Interactions Between RPL and Radio Duty Cycling Protocols Affect QoS in Wireless Sensor Networks. In *Proceedings of the 12th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet '16*, pages 135–138, New York, NY, USA. ACM.
- [15] Bieszczad, A., Pagurek, B., and White, T. (1998). Mobile agents for network management. *IEEE Communications Surveys*, 1(1):2–9.
- [16] Boukerche, A., Loureiro, A. A., Nakamura, E. F., Oliveira, H. A., Ramos, H. S., and Villas, L. A. (2014). Cloud-assisted computing for event-driven mobile services. *Mobile Networks and Applications*, 19.
- [17] C., P., S., R., and J., D. (2013). Dynamic MANET On-demand (AODVv2) Routing. <https://tools.ietf.org/html/draft-ietf-manet-dymo-26>.
- [18] Camp, T., Boleng, J., and Davies, V. (2002). A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502.
- [19] Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., and Scott, J. (2005). Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. Technical report, University of Cambridge, Computer Laboratory.
- [20] Chan, H., Liu, D., Seite, P., Yokota, H., and Korhonen, J. (2014). Requirements for distributed mobility management.
- [21] Chlipala, A., Hui, J., and Tolle, G. (2004). Deluge: data dissemination for network reprogramming at scale. *University of California*.

- [22] Choi, Y., Hong, Y.-G., Youn, J.-S., Kim, D., and Choi, J. (2018). Transmission of IPv6 Packets over Near Field Communication. Internet-Draft draft-ietf-6lo-nfc-12, Internet Engineering Task Force. Work in Progress.
- [23] Clark, B. N., Colbourn, C. J., and Johnson, D. S. (1991). Unit Disk Graphs. *Discrete mathematics*, 86(1-3):165–177.
- [24] Clausen, T. and Jacquet, P. (2003). Optimized link state routing protocol (olsr). RFC 3626.
- [25] Cobarzan, C., Montavont, J., and Noel, T. (2014). Analysis and performance evaluation of RPL under mobility. In *IEEE ISCC*, pages 1–6.
- [26] Cunha, F., Maia, G., Ramos, H., Santos, B. P., Celes, C., Rettore, P., Campolina, A., Guidoni, D., Souza, F., Villas, L., Mini, R., and Loureiro., A. (2017a). *Emerging Trends in Vehicular Ad-hoc Network (VANET)*. Emerging Wireless Communication and Network Technologies: Principle, Paradigm and Performance. Springer Chapter Book.
- [27] Cunha, F., Maia, G., Ramos, H., Santos, B. P., Celes, C., Rettore, P., Campolina, A., Guidoni, D., Souza, F., Villas, L., Mini, R., and Loureiro., A. (2017b). *Sistemas de Transporte Inteligentes: Conceitos, Aplicações e Desafios (Intelligent Transportation Systems: Concepts, Applications and Challenges - In portuguese)*. XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. 1ed.Porto Alegre: Sociedade Brasileira de Computação (SBC) SBRC 2017. Chapter Book.
- [28] Da Xu, L., He, W., and Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243.
- [29] Dawson-Haggerty, S., Tavakoli, A., and Culler, D. (2010a). Hydro: A hybrid routing protocol for low-power and lossy networks. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 268–273. IEEE.
- [30] Dawson-Haggerty, S., Tavakoli, A., and Culler, D. (2010b). Hydro: A hybrid routing protocol for low-power and lossy networks. In *IEEE SmartGridComm*, pages 268–273.
- [31] de Souza, F., Domingues, A. C., Vaz de Melo, P., and A.F. Loureiro, A. (2018). MOCHA: A tool for mobility characterization. In *MSWiM*.

- [32] Deering, D. S. E. and Hinden, B. (2017). Internet Protocol, Version 6 (IPv6) Specification. RFC 8200.
- [33] Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *IEEE LCN*, pages 455–462, Washington, DC, USA. IEEE Computer Society.
- [34] Duquennoy, S., Landsiedel, O., and Voigt, T. (2013). Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 2:1–2:14, New York, NY, USA. ACM.
- [35] Eagle, N. and Pentland, A. S. (2006). Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268.
- [36] Ekman, F., Keränen, A., Karvo, J., and Ott, J. (2008). Working day movement model. In *Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models*, pages 33–40. ACM.
- [37] El Korbi, I., Brahim, M. B., Adjih, C., and Saidane, L. A. (2012). Mobility enhanced RPL for wireless sensor networks. In *IEEE ICUFN*, pages 1–8.
- [38] Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., and Marrón, P. J. (2009). COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks. In *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, Simutools'09, pages 27:1–27:7.
- [39] Farinacc, D., Fuller, V., Meyer, D., and Lewis, D. (2013). RFC 6830 - The Locator/ID Separation Protocol (LISP). <https://tools.ietf.org/html/rfc6830>. (Accessed on 04/01/2018).
- [40] Fonseca, R., Gnawali, O., Jamieson, K., and Levis, P. (2007). Four-Bit Wireless Link Estimation. In *HotNets*.
- [41] Foroozani, A., Gharib, M., Hemmatyar, A. M. A., and Movaghar, A. (2014). A novel human mobility model for manets based on real data. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7. IEEE.
- [42] Fotouhi, H., Moreira, D., and Alves, M. (2015). mRPL: Boosting mobility in the Internet of Things. *Ad Hoc Networks*, pages 17–35.

- [43] Gaddour, O., Koubâa, A., Rangarajan, R., Cheikhrouhou, O., Tovar, E., and Abid, M. (2014). Co-RPL: RPL routing for mobile low power wireless sensor networks using Corona mechanism. In *Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on*, pages 200–209. IEEE.
- [44] Gara, F., Ben Saad, L., Ben Hamida, E., Tourancheau, B., and Ben Ayed, R. (2016). An adaptive timer for RPL to handle mobility in wireless sensor networks. *IWCMC 2016*, (978):678–683.
- [45] Gartner, I. (2018). Hype Cycle Research Methodology | Gartner Inc. <https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp>. (Accessed on 03/22/2018).
- [46] Gershenfeld, N. (1999). *When Things Start to Think*. Henry Holt and Co., Inc., New York, NY, USA.
- [47] Gnawali, O., Fonseca, R., Jamieson, K., Kazandjjeva, M., Moss, D., and Levis, P. (2013). CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks. *ACM TOSN*, 10(1):16.
- [48] Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., and Levis, P. (2009). Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, SenSys '09*, pages 1–14.
- [49] Group, E. D. C. (2008). Sinalgo-simulator for network algorithms.
- [50] Guo, B., Yu, Z., Zhou, X., and Zhang, D. (2012). Opportunistic IoT: Exploring the social side of the internet of things. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, pages 925–929. IEEE.
- [51] Guo, S., Wang, M., and Leskovec, J. (2011). The role of social networks in online shopping: information passing, price of trust, and consumer choice. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 157–166. ACM.
- [52] Hassan, A., Ahmed, M. H., and Rahman, M. A. (2013). Adaptive beaconing system based on fuzzy logic approach for vehicular network. *IEEE PIMRC*, pages 2581–2585.
- [53] Hess, A., Hummel, K. A., Gansterer, W. N., and Haring, G. (2016). Data-driven human mobility modeling: A survey and engineering guidance for mobile networking. *ACM Computing Surveys (CSUR)*, 48(3):38.

- [54] Hong, K.-S. and Choi, L. (2011). DAG-based multipath routing for mobile sensor networks. In *IEEE ICT*, pages 261–266.
- [55] Hong, X., Gerla, M., Pei, G., and Chiang, C.-C. (1999). A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60. ACM.
- [56] Hui, P., Crowcroft, J., and Yoneki, E. (2011). Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576–1589.
- [57] Iova, O., Picco, P., Istomin, T., and Kiraly, C. (2016). RPL: The Routing Standard for the Internet of Things... Or Is It? *IEEE Communications Magazine*, 54:16–22.
- [58] Johnson, D., Hu, Y., Maltz, D., et al. (2007). The dynamic source routing protocol for mobile ad hoc networks. Technical report, RFC 4728.
- [59] Júnior, N. d. S. R., Vieira, M. A. M., Vieira, L. F. M., and Gnawali, O. (2014). CodeDrip: Data dissemination protocol with network coding for wireless sensor networks. In *Wireless Sensor Networks*, pages 34–49. Springer.
- [60] Karamshuk, D., Boldrini, C., Conti, M., and Passarella, A. (2011). Human mobility models for opportunistic networks. *IEEE Communications Magazine*, 49(12):157–165.
- [61] Karamshuk, D., Boldrini, C., Conti, M., and Passarella, A. (2014). SPoT: Representing the social, spatial, and temporal dimensions of human mobility with a unifying framework. *Pervasive and Mobile Computing*, 11:19–40.
- [62] Kelly, S. D. T., Suryadevara, N. K., and Mukhopadhyay, S. C. (2013). Towards the implementation of IoT for environmental condition monitoring in homes. *IEEE Sensors Journal*, 13(10):3846–3853.
- [63] Kim, K., Montenegro, G., Park, S., Chakeres, I., and Perkins, C. (2007a). Dynamic MANET On-demand for 6LoWPAN (DYMO-low) Routing. *Internet Engineering Task Force*.
- [64] Kim, K., Park, S. D., Montenegro, G., Yoo, S., and Kushalnagar, N. (2007b). 6LoWPAN ad hoc on-demand distance vector routing (LOAD). *Network WG Internet Draft*, 19.

- [65] Kosta, S., Mei, A., and Stefa, J. (2012). Large-scale synthetic social mobile networks with SWIM. *IEEE Transactions on Mobile Computing*, 13(1):116–129.
- [66] Kotz, D. and Henderson, T. (2005). Crawdad: A community resource for archiving wireless data at dartmouth. *IEEE Pervasive Computing*, 4(4):12–14.
- [67] Kurose, J. F. and Ross, K. W. (2012). *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 6th edition.
- [68] Kushalnagar, N., Montenegro, G., and Schumacher, C. (2007). IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals. Technical report.
- [69] Landt, J. (2005). The history of RFID. *IEEE potentials*, 24(4):8–11.
- [70] Lee, K. C., Sudhaakar, R., Dai, L., Addepalli, S., and Gerla, M. (2012). RPL under mobility. In *IEEE CCNC*, pages 300–304. IEEE.
- [71] Leppanen, T., Liu, M., Harjula, E., Ramalingam, A., Ylioja, J., Narhi, P., Riekkii, J., and Ojala, T. (2013). Mobile agents for integration of internet of things and wireless sensor networks. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 14–21. IEEE.
- [72] Levis, P., Lee, N., Welsh, M., and Culler, D. (2003). TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03*, pages 126–137, New York, NY, USA. ACM.
- [73] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., et al. (2005a). TinyOS: An operating system for sensor networks. *Ambient intelligence*, 35:115–148.
- [74] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., et al. (2005b). TinyOS: An operating system for sensor networks. *Ambient intelligence*, 35:115–148.
- [75] Levis, P., Patel, N., Culler, D., and Shenker, S. (2004). Trickle: A Self-regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1, NSDI'04*, pages 2–2.

- [76] Libelium (2013). Libelium Smart World Infographic – Sensors for Smart Cities, Internet of Things and beyond | Libelium. <http://www.libelium.com/libelium-smart-world-infographic-smart-cities-internet-of-things/>. (Accessed on 09/05/2018).
- [77] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., and Zhao, W. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142.
- [78] Lodhi, M. A., Rehman, A., Khan, M. M., and Hussain, F. B. (2015). Multiple path RPL for low power lossy networks. In *Wireless and Mobile (APWiMob), 2015 IEEE Asia Pacific Conference on*, pages 279–284.
- [79] Loureiro, A. A., Nogueira, J. M. S., Ruiz, L. B., Mini, R. A. d. F., Nakamura, E. F., and Figueiredo, C. M. S. (2003). Redes de Sensores Sem Fio. In *Simpósio Brasileiro de Redes de Computadores (SBRC)*, pages 179–226.
- [80] Mattern, F. (2000). Mobile Agents. <https://www.vs.inf.ethz.ch/publ/slides/MobAgsTut.pdf>. (Accessed on 04/01/2018).
- [81] Mei, A. and Stefa, J. (2009). SWIM: A Simple Model to Generate Small Mobile Worlds. In *IEEE INFOCOM 2009*, pages 2106–2113.
- [82] Moghadam, M. N., Taheri, H., and Karrari, M. (2015). Multi-class Multipath Routing Protocol for Low Power Wireless Networks with Heuristic Optimal Load Distribution. *Wireless Personal Communications*, 82(2):861–881.
- [83] Montenegro, G., Schumacher, C., and Kushalnagar, N. (2007). IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919.
- [84] Mota, V. F., Cunha, F. D., Macedo, D. F., Nogueira, J. M., and Loureiro, A. A. (2014). Protocols, mobility models and tools in opportunistic networks: A survey. *Computer Communications*, 48:5–19.
- [85] Mulligan, G. (2007). The 6lowpan architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 78–82. ACM.
- [86] Nahrstedt, K., Li, H., Nguyen, P., Chang, S., and Vu, L. (2016). Internet of mobile things: Mobility-driven challenges, designs and implementations. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pages 25–36. IEEE.

- [87] Narendra, N. and Misra, P. (2016). Research challenges in the internet of mobile things.
- [88] Ng, A., Ngiam, J., Foo, C. Y., Mai, Y., and Suen, C. (2011). Backpropagation Algorithm. http://ufldl.stanford.edu/wiki/index.php/Backpropagation_Algorithm. (Accessed on 10/11/2018).
- [89] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and Gomez, C. (2015). IPv6 over BLUETOOTH(R) Low Energy. RFC 7668.
- [90] Nordrum, A. (2016). Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated - IEEE Spectrum. <https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated> (Accessed on 03/22/2018).
- [91] Nunes, I. O., Celes, C., Silva, M. D., Vaz de Melo, P. O., and Loureiro, A. A. (2017). GRM: Group Regularity Mobility Model. In *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '17, pages 85–89, New York, NY, USA. ACM.
- [92] Oliveira, A. and Vazão, T. (2016). Low-power and lossy networks under mobility: A survey. *Computer Networks*, 107:339–352.
- [93] Palani, U., Alamelumangai, V., and Nachiappan, A. (2015). Hybrid routing and load balancing protocol for wireless sensor network. *Wireless Networks*, pages 1–8.
- [94] Peres, B., Santos, B. P., de O. Souza, O. A., Goussevskaia, O., Vieira, M. A. M., Vieira, L. F. M., and Loureiro, A. A. F. (2018). Matrix: Multihop Address allocation and dynamic any-To-any Routing for 6LoWPAN. *Computer Networks*, 140:28 – 40.
- [95] Peres, B. S., Souza, O. A. d. O., Santos, B. P., Junior, E. R. A., Goussevskaia, O., Vieira, M. A. M., Vieira, L. F. M., and Loureiro, A. A. F. (2016). Matrix: Multihop Address Allocation and Dynamic Any-to-Any Routing for 6LoWPAN. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 302–309, New York, NY, USA. ACM.
- [96] Perkins, C., Johnson, D., and Arkko, J. (2011). Mobility support in IPv6. RFC 6275.

- [97] Perkins, C. E. and Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*.
- [98] Peterson, L. L. and Davie, B. S. (2011). *Computer Networks, Fifth Edition: A Systems Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.
- [99] Qadori, H. Q., Zulkarnain, Z. A., Hanapi, Z. M., and Subramaniam, S. (2017). Multi-mobile agent itinerary planning algorithms for data gathering in wireless sensor networks: A review paper. *International Journal of Distributed Sensor Networks*, 13(1):1550147716684841.
- [100] Qiu, F., Zhou, H., Li, X., Wang, G., and Zhang, H. (2014). A distributed mobility management scheme in networks with the locator/identifier separation. *International Journal of Communication Systems*, 27(10):1874–1893.
- [101] Reinhardt, A., Morar, O., Santini, S., Zoller, S., and Steinmetz, R. (2012). CBFR: Bloom filter routing with gradual forgetting for tree-structured wireless sensor networks with mobile nodes. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–9.
- [102] Rettore, P. H., Santos, B. P., Campolina, A. B., Villas, L. A., and Loureiro, A. A. (2016). Towards intra-vehicular sensor data fusion. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 126–131. IEEE.
- [103] Rettore, P. H., Santos, B. P., Maia, G., Villas, L. A., and Loureiro, A. A. F. (2018). Rode: Road data enrichment framework based on heterogeneous data fusion for its. *Transactions on Intelligent Transportation Systems*.
- [104] Royer, E. M. and Toh, C.-K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. *IEEE personal communications*, 6(2):46–55.
- [105] Ruiz, L. B., Correia, L. H. A., Vieira, L. F. M., Macedo, D. F., Nakamura, E. F., Figueiredo, C. M., Vieira, M. A. M., Bechelane, E. H., Camara, D., Loureiro, A. A., et al. (2004). Arquiteturas para Redes de Sensores Sem Fio. In *Simpósio Brasileiro de Redes de Computadores (SBRC)*.
- [106] Santos, B. P., Goussevskaia, O., Vieira, L. F., Vieira, M. A., and Loureiro, A. A. (2018a). Mobile Matrix: Routing under Mobility in IoT, IoMT, and Social IoT. *Ad Hoc Networks*.

- [107] Santos, B. P., Goussevskaia, O., Vieira, L. F. M., Vieira, M. A. M., and Loureiro, A. A. (2017a). Mobile Matrix: A Multihop Address Allocation and Any-to-Any Routing in Mobile 6LoWPAN. In *Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet '17*, pages 65–72.
- [108] Santos, B. P., Rettore, P. H., Ramos, H. S., Vieira, L. F., and Loureiro, A. A. (2017b). T-MAPS: Modelo de Descrição do Cenário de Trânsito Baseado no Twitter. *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.
- [109] Santos, B. P., Rettore, P. H., Ramos, H. S., Vieira, L. F., and Loureiro, A. A. (2018b). Enriching Traffic Information with a Spatiotemporal Model based on Social Media. In *Computers and Communications (ISCC), 2018 IEEE Symposium on*, page 0. IEEE.
- [110] Santos, B. P., Rettore, P. H. L., Vieira, L. F. M., and Loureiro, A. A. (2018c). Dribble: a learn-based timer scheme selector for mobility management in iot. *IEEE Wireless Communications and Networking Conference WCNC*.
- [111] Santos, B. P., Silva, L., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. (2016). Internet das coisas: da teoria à prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.
- [112] Santos, B. P., Vieira, L. F. M., and Vieira, M. A. M. (2017c). CGR: Centrality-based green routing for Low-power and Lossy Networks. *Computer Networks*, 129:117–128.
- [113] Santos, B. P., Vieira, M. A. M., and Vieira, L. F. M. (2015). eXtend collection tree protocol. In *Wireless Communications and Networking Conference (WCNC), 2015 IEEE*, pages 1512–1517. IEEE.
- [114] Sattlegger, K. and Denk, U. (2014). Navigating your way through the RFID jungle. *White paper, Texas Instruments*.
- [115] Saucez, D., Iannone, L., Bonaventure, O., and Farinacci, D. (2012). Designing a deployable internet: The locator/identifier separation protocol. *IEEE Internet Computing*, 16(6):14–21.
- [116] Schmidt, R. K., Leinmuller, T., Schoch, E., Kargl, F., and Schafer, G. (2010). Exploration of adaptive beaconing for efficient intervehicle safety communication. *IEEE Network*, 24(1):14–19.

- [117] Shelby, Z. and Bormann, C. (2011). *6LoWPAN: The wireless embedded Internet*, volume 43. John Wiley & Sons.
- [118] Silva, T. H., de Melo, P. O. V., Almeida, J. M., Musolesi, M., and Loureiro, A. A. (2017). A large-scale study of cultural differences using urban data about eating and drinking preferences. *Information Systems*, 72:95–116.
- [119] Sommer, C., Tonguz, O. K., and Dressler, F. (2010). Adaptive beaconing for delay-sensitive and congestion-aware traffic information systems. In *IEEE VNC*.
- [120] Talavera, L. E., Endler, M., Vasconcelos, I., Vasconcelos, R., Cunha, M., and e Silva, F. J. d. S. (2015). The mobile hub concept: Enabling applications for the internet of mobile things. In *Pervasive computing and communication workshops (PerCom workshops), 2015 IEEE international conference on*, pages 123–128. IEEE.
- [121] Tanenbaum, A. (2002). *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition.
- [122] Thubert, P. and Hui, J. (2011). Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282.
- [123] Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D., and Hong, W. (2005). A Macroscope in the Redwoods. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05*, pages 51–63.
- [124] University, C. M. (2014). The "Only" Coke Machine on the Internet. https://www.cs.cmu.edu/~coke/history_long.txt. (Accessed on 09/05/2018).
- [125] Vasseur, J.-P. and Dunkels, A. (2010). *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann.
- [126] Vicaire, P., He, T., Cao, Q., Yan, T., Zhou, G., Gu, L., Luo, L., Stoleru, R., Stankovic, J. A., and Abdelzaher, T. F. (2009). Achieving Long-term Surveillance in VigilNet. *ACM Transactions on Sensor Networks (TOSN)*, 5(1):9:1–9:39.
- [127] Voutilainen, J.-P., Mattila, A.-L., Systä, K., and Mikkonen, T. (2016). HTML5-based mobile agents for Web-of-Things. *Informatica*, 40(1):43.
- [128] Weis, S. A. (2007). Rfid (radio frequency identification): Principles and applications. *System*, 2(3):1–23.

- [129] Weiser, M. (1999). The computer for the 21st century. *Mobile Computing and Communications Review*, 3(3):3–11.
- [130] Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. (2006a). Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 381–396.
- [131] Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. (2006b). Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 381–396. USENIX Association.
- [132] Whitbeck, J., de Amorim, M. D., and Conan, V. (2010). Plausible Mobility: Inferring Movement from Contacts. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, pages 110–117, New York, NY, USA. ACM.
- [133] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard).
- [134] Woo, A., Tong, T., and Culler, D. (2003). Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *International Conference on Embedded Networked Sensor Systems*. ACM.
- [135] Zheng, Y., Zhang, L., Xie, X., and Ma, W.-Y. (2009). Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM.
- [136] Zignani, M. and Gaito, S. (2010). Extracting human mobility patterns from gps-based traces. In *2010 IFIP Wireless Days*, pages 1–5. IEEE.

