Knowledge driven Discovery for Opportunistic IoT Networking

Riccardo Pozza

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

Institute for Communication Systems
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, UK

July 2015

© Riccardo Pozza, 2015
Summary

So far, the Internet of Things (IoT) has been concerned with the objective of connecting everything, or any object to the Internet world. By collaborating towards the creation of new services, the IoT has introduced the opportunity to add smartness to our cities, homes, buildings and healthcare systems, as well as businesses and products. In many scenarios, objects or IoT devices are not always statically deployed, but they may be free to move around being carried by people or vehicles, while still interacting with static IoT infrastructure. The Opportunistic Networking paradigm states that, exploiting opportunistic interactions between static and mobile IoT devices, provides for increased network capacity, additional connectivity, reduced deployment costs, improved reliability and overall network lifetime improvements.

IoT scenarios do illustrate the increased need to identify and exploit opportunistic interactions between IoT devices in order to recognize when an opportunity for communication is possible. For example, statically deployed devices (i.e. road side sensors) may need to find mobile devices (this may be sensors or actuators) (i.e. connected cars) for exploiting them for collecting and relaying data towards destinations without relying on a static infrastructure. This means that discovery in IoT scenarios needs to determine the availability of other devices in scenarios in which devices’ presence is uncertain or may change over time. This directly leads to a contradicting objective where resource wastage in device discovery is to be kept at a minimum.

This thesis presents two contributions that provide solutions to overcome the clash between these contradicting objectives. Firstly, a Context Aware Resource Discovery mechanism is introduced, capable of providing optimized discovery and adapting available resources based on learned mobility patterns. Secondly, an Arrival and Departure Time Prediction and Discovery framework is defined and investigated; this framework aims to predict future arrival and departure times and helps to plan the use of devices’ resources in advance based on the foreseen resource demand patterns.

Keywords: Internet of Things, Machine Learning, Mobility, Knowledge, Discovery

Email: r.pozza@surrey.ac.uk
WWW: http://www.surrey.ac.uk/ics/
Acknowledgements

Many people deserve my gratitude and are in part responsible for my achievements over the last few years spent during my PhD.

First of all, I would like to thank my supervisors for their guidance and support over the years I’ve spent working towards my research objectives. Without their experience and their ready availability to support my research with suggestions and recommendations, it would not have been possible to overcome the difficulties encountered during these past years.

In addition, I would like to thank all the friends and people in the centre that supported me either directly or indirectly always with a friendly word or with a piece of advice.

Finally, I would like to thank my family and all of my friends, without whose support over these years, all that I’ve achieved would not have been possible.

A special thank goes to my other half, whose support and patience over all of these years, especially when I needed to work for long hours, made it less difficult.
Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>Glossary of Terms</td>
<td>ix</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 The Challenges</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1 Learning and Extracting Knowledge about Mobility Patterns</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Resource Scheduling for Optimized, Low Latency and Energy-Efficient Discovery</td>
<td>7</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>9</td>
</tr>
<tr>
<td>1.4 Organisation</td>
<td>11</td>
</tr>
<tr>
<td>2 Background and Related Work</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Neighbour Discovery for Opportunistic Networking in IoT scenarios</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Mobility Agnostic Discovery Protocols</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1 Time Synchronized Protocols</td>
<td>17</td>
</tr>
<tr>
<td>2.2.2 Asynchronous Protocols</td>
<td>18</td>
</tr>
<tr>
<td>2.3 Mobility Driven Discovery Protocols</td>
<td>23</td>
</tr>
<tr>
<td>2.3.1 Temporal Knowledge Based</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2 Spatial Knowledge Based</td>
<td>25</td>
</tr>
<tr>
<td>2.4 Shortcomings and Discussion</td>
<td>27</td>
</tr>
<tr>
<td>2.4.1 Contribution to State-of-the-art</td>
<td>29</td>
</tr>
<tr>
<td>3 Context Aware Resource Discovery</td>
<td>33</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>33</td>
</tr>
</tbody>
</table>
Contents

3.2 Current discovery approaches analysis and discussion .......................... 34
   3.2.1 Disco .............................................. 35
   3.2.2 Reinforcement Learning and Q-Learning ................................. 38
   3.2.3 RADA ............................................. 40
3.3 Proposed System Model for Context Aware Discovery ..................... 41
3.4 Learning Model for Context Aware Discovery ................................. 43
   3.4.1 Disco-based Schedule Model ..................................... 44
   3.4.2 CARD Actions Model ....................................... 45
   3.4.3 CARD States Model ......................................... 46
   3.4.4 CARD Actions Schedule Parameters ................................ 47
   3.4.5 CARD Reward Function Model ................................... 48
   3.4.6 CARD Learning and Additional Parameters ........................... 51
3.5 Conclusions ............................................................................ 51

4 Arrival and Departure Time Prediction and Discovery .................... 53
   4.1 Introduction ........................................................................ 53
   4.2 Learning Algorithms Based on Temporal Differences Methods ...... 54
      4.2.1 Function Approximation and Least Squares Temporal Difference Methods 57
   4.3 Arrivals and Departures Prediction Algorithm ........................... 59
   4.4 Resource Scheduling based on Next Contact Predictions ............. 62
   4.5 Conclusions ....................................................................... 66

5 Implementation ............................................................................ 69
   5.1 Introduction ......................................................................... 69
      5.1.1 Network Simulator Overview and Extensions ..................... 70
   5.2 Resource Aware Data Accumulation ...................................... 77
   5.3 Context Aware Resource Discovery ...................................... 79
   5.4 Arrival and Departure Time Predictor .................................... 81
   5.5 Arrival and Departure Time Scheduler Planner ....................... 84
   5.6 Conclusions ....................................................................... 86

6 Evaluation ..................................................................................... 87
   6.1 Introduction ......................................................................... 87
   6.2 RADA Validation .................................................................. 88
   6.3 CARD Performance Evaluation .......................................... 91
   6.4 ADTP Predictor Evaluation ................................................ 100
   6.5 ADTP Discovery Planner Evaluation ..................................... 112
   6.6 Conclusions ....................................................................... 126

7 Conclusions .................................................................................. 129
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Mobility Pattern Features</td>
<td>5</td>
</tr>
<tr>
<td>1.2</td>
<td>Learning, Planning and Discovery Optimization</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>IoT scenario of Opportunistic Networking</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Main areas of research in Neighbour Discovery for Opportunistic Networking in IoT scenarios</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>Time Synchronized Discovery</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Indirect Request Driven Discovery</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Temporal Overlap Discovery</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Temporal Knowledge Based Discovery</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>Spatial Knowledge Based Discovery</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Disco between IoT devices $i$ and $j$ with coprime pair $m_i = 5$ and $m_j = 2$</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>A sample IoT scenario of Opportunistic Networking</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td>Temporal parameters used in CARD</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>CARD $A(4,1)$ action with $N_S = 6$</td>
<td>46</td>
</tr>
<tr>
<td>3.5</td>
<td>CARD $S(1,1)$ state reached after the $A(4,1)$ action with $N_S = 6$</td>
<td>46</td>
</tr>
<tr>
<td>3.6</td>
<td>CARD reward in $S(1,1)$ state reached after the $A(4,1)$ action with $N_S = 6$</td>
<td>50</td>
</tr>
<tr>
<td>3.7</td>
<td>CARD reward in $S(1,1)$ state reached after the $A(4,1)$ action with $N_S = 6$</td>
<td>50</td>
</tr>
<tr>
<td>4.1</td>
<td>Prediction with Temporal Difference Learning</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>ADTP Resource Scheduler</td>
<td>63</td>
</tr>
<tr>
<td>4.3</td>
<td>ADTP Resource Schedule</td>
<td>65</td>
</tr>
<tr>
<td>5.1</td>
<td>NS-3 Simulator Modules</td>
<td>71</td>
</tr>
<tr>
<td>5.2</td>
<td>NS-3 Networking Stack</td>
<td>72</td>
</tr>
<tr>
<td>5.3</td>
<td>NS-3 Energy Model</td>
<td>74</td>
</tr>
<tr>
<td>5.4</td>
<td>Synthetic Mobility Traces</td>
<td>75</td>
</tr>
<tr>
<td>5.5</td>
<td>Resource Aware Data Accumulation Application</td>
<td>78</td>
</tr>
<tr>
<td>5.6</td>
<td>Context Aware Resource Discovery Application</td>
<td>80</td>
</tr>
<tr>
<td>5.7</td>
<td>PyBrain Reinforcement Learning Framework</td>
<td>82</td>
</tr>
</tbody>
</table>
5.8 Arrival and Departure Time Prediction Main Program. . . . . . . . . . . . . . 83
5.9 Arrival and Departure Time Prediction Application. . . . . . . . . . . . . . 85

6.1 Number of Task executions and Exploration Strategy at 3.6Km/h. . . . . . . . 90
6.2 Number of Task executions and Exploration Strategy at 40Km/h. . . . . . . . 90
6.3 LogDistance and Nakagami-m Fast Fading loss models for a MICA2 node. . . 91
6.4 Total Cumulative Residual Contact Time (Deterministic). . . . . . . . . . . . . 94
6.5 Energy Breakdown (Deterministic). . . . . . . . . . . . . . . . . . . . . . . . . . 94
6.6 Average Latency (Deterministic). . . . . . . . . . . . . . . . . . . . . . . . . . . 95
6.7 Total Cumulative Residual Contact Time (Multiple Deterministic). . . . . . . 96
6.8 Energy Breakdown (Multiple Deterministic). . . . . . . . . . . . . . . . . . . . 96
6.9 Average Latency (Multiple Deterministic). . . . . . . . . . . . . . . . . . . . . 97
6.10 Energy Breakdown (Gaussian). . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
6.11 Total Cumulative Residual Contact Time. . . . . . . . . . . . . . . . . . . . . 98
6.12 Average Latency. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 98
6.13 Energy per second of useful contact discovered. . . . . . . . . . . . . . . . . . . 99
6.14 Deterministic Mobility Pattern. . . . . . . . . . . . . . . . . . . . . . . . . . . . 102
6.15 Multiple Deterministic Mobility Pattern. . . . . . . . . . . . . . . . . . . . . . 103
6.16 Gaussian Mobility Pattern. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
6.17 Multiple Gaussian Mobility Pattern. . . . . . . . . . . . . . . . . . . . . . . . . 106
6.18 Bluetooth Traces. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
6.19 P.I.R. Traces. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108
6.20 Intel Traces. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 109
6.21 Cambridge Traces. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
6.22 Infocom 2005 Traces. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112
6.23 Deterministic Mobility Scenario Results. . . . . . . . . . . . . . . . . . . . . . . 115
6.24 Multiple Deterministic Mobility Scenario Results. . . . . . . . . . . . . . . . . 116
6.25 Gaussian Mobility Scenario Results. . . . . . . . . . . . . . . . . . . . . . . . . 118
6.26 Multiple Gaussian Mobility Scenario Results. . . . . . . . . . . . . . . . . . . 119
6.27 Bluetooth Trace Mobility Scenario Results. . . . . . . . . . . . . . . . . . . . 120
6.28 P.I.R. Trace Mobility Scenario Results. . . . . . . . . . . . . . . . . . . . . . . 121
6.29 Intel Trace Mobility Scenario Results. . . . . . . . . . . . . . . . . . . . . . . . 122
6.30 Cambridge Trace Mobility Scenario Results. . . . . . . . . . . . . . . . . . . 123
6.31 Infocom Trace Mobility Scenario Results. . . . . . . . . . . . . . . . . . . . . . 125
6.32 STEPS Mobility Model Results. . . . . . . . . . . . . . . . . . . . . . . . . . . 126
# Glossary of Terms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADTP</td>
<td>Arrival and Departure Time Prediction</td>
</tr>
<tr>
<td>AEB</td>
<td>Adaptive Exponential Beacon</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Networks</td>
</tr>
<tr>
<td>AODV</td>
<td>Ad hoc On-Demand Distance Vector</td>
</tr>
<tr>
<td>AP</td>
<td>Access Point</td>
</tr>
<tr>
<td>AQEC</td>
<td>Adaptive Quorum-based Energy Conserving</td>
</tr>
<tr>
<td>ARP</td>
<td>Address Resolution Protocol</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat reQuest</td>
</tr>
<tr>
<td>BAN</td>
<td>Body Area Networks</td>
</tr>
<tr>
<td>BAW</td>
<td>Bulk Acoustic Wave</td>
</tr>
<tr>
<td>CAPM</td>
<td>Context Aware Power Management</td>
</tr>
<tr>
<td>CARD</td>
<td>Context Aware Resource Discovery</td>
</tr>
<tr>
<td>CCA</td>
<td>Clear Channel Assessment</td>
</tr>
<tr>
<td>CCDF</td>
<td>Complementary Cumulative Distribution Function</td>
</tr>
<tr>
<td>CDC</td>
<td>Cooperative Duty Cycling</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>CenWits</td>
<td>Connection-less Sensor-Based Tracking System Using Witnesses</td>
</tr>
<tr>
<td>DSDV</td>
<td>Destination-sequenced Distance Vector</td>
</tr>
<tr>
<td>DSR</td>
<td>Dynamic Source Routing</td>
</tr>
<tr>
<td>DTN</td>
<td>Delay Tolerant Networks</td>
</tr>
<tr>
<td>GeRaF</td>
<td>Geographic Random Forwarding</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IPv4</td>
<td>Internet Protocol version 4</td>
</tr>
<tr>
<td>IPv6</td>
<td>Internet Protocol version 6</td>
</tr>
<tr>
<td>LSTD</td>
<td>Least Squares Temporal Difference</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad Hoc Network</td>
</tr>
<tr>
<td>MAUC</td>
<td>Mobility Assisted User Contact</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov Decision Process</td>
</tr>
<tr>
<td>NFC</td>
<td>Near Field Communications</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NIx-Vector</td>
<td>Neighbour-Index Vector</td>
</tr>
<tr>
<td>NS-3</td>
<td>Network Simulator 3</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>OLSR</td>
<td>Optimized Link State Routing</td>
</tr>
<tr>
<td>OOK</td>
<td>On-Off Keying</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>PyBrain</td>
<td>Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network</td>
</tr>
<tr>
<td>RADA</td>
<td>Resource Aware Data Accumulation</td>
</tr>
<tr>
<td>RAW</td>
<td>Random Asynchronous Wakeup</td>
</tr>
<tr>
<td>RBTP</td>
<td>Recursive Binary Time Partitioning</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency IDentification</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indication</td>
</tr>
<tr>
<td>SAW</td>
<td>Surface Acoustic Wave</td>
</tr>
<tr>
<td>SNIP-RH</td>
<td>Sensor Node Initiated Probing for Rush Hours</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>STEM</td>
<td>Sparse Topology Energy Management</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TD</td>
<td>Temporal Differences</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>WiSaG</td>
<td>Wi-Fi Sensing with aGing</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Networks</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter provides an introduction to the research problem this thesis is engaging with, as well as an overview of the recent challenges in neighbour discovery when IoT scenarios of opportunistic networking between devices are considered. A list of the principal research contributions is reported, along with the structure of this thesis.

1.1 Overview

The Internet of Things (IoT) is a concept pursuing the objective of expanding the reach of the current Internet world towards any possible object or thing that needs to be connected. Such a paradigm has its foundations in the experience gathered by the research community in wireless networking through their studies on Near Field Communications (NFC) and Wireless Sensor Networks (WSN). In such a world, Radio Frequency Identification (RFID) tags as well as sensors and actuators with wireless radios, allow the exchange of information between real world objects.

One of the principal aims of the IoT is to make possible to foster the cooperation and collaboration between objects in order to devise new “smart” services and applications. Notable examples thereof include Smart Cities, Smart Homes and Buildings, Environmental Monitoring, Healthcare, Smart Business, Inventory and Product Management, as well as Security and Surveillance. In order to pursue such an objective, the IoT needs a common architecture where smart and efficient networking protocols are required to bridge the gap between pervasive communication needs and devices’ lifetime in scenarios where mobility plays a fundamental role. Indeed, it has been shown that mobility can not only increase network capacity, but also provide additional connectivity for (partially disconnected) sparse networks. Moreover, it has been also shown that mobility can improve reliability and energy efficiency by providing shorter paths in comparison with traditional static sink-based networks. In addition, one of the driving factors for IoT scenarios is that the exploitation of mobility allows for increasing the
connectivity to the Internet world by opportunistically communicating between devices that might be momentarily disconnected.

It is evident that the applications aforementioned need to handle contacts between IoT devices which, from a temporal point of view, might be rare and short. This would require the exploitation of any opportunity to communicate in its entirety without the possibility to rely on a deployed backbone infrastructure. For example, in the case of a node deployed in a field, but close to a road, vehicles in the neighbourhood could opportunistically allow agriculture data collection. Opportunistic Networking [7] indeed envisions that freely moving (mobile) devices might opportunistically interact with each other and with fixed position (static) devices, in order to collect or disseminate data and to offload computation or to enable forwarding and routing between any kind of these devices [8]. For example, in a Smart City, where sensing and actuating devices are deployed, vehicles or human carried devices could opportunistically collect data from devices or provide higher computational capabilities to them, as well as relaying the collected data towards the intended destination. This could avoid relying on a deployed static infrastructure for running their applications. Examples thereof are: vehicle traffic monitoring, garbage collection management, environmental data collection, etc.

From a networking point of view, guaranteeing maximum device lifetime and optimal useful time for communication between devices in such an IoT scenario are very challenging and contradicting objectives. In fact, while frequent energy resource scheduling operations could increase the likeliness to find a device in the vicinity, this considerably reduces device energy reserve. In addition, an important part of device energy would be wasted by searching for neighbours which might not be present at all, even for long periods of time. Moreover, an increasing number of researchers in the community have shown interest in developing an understanding of mobility patterns (i.e. of human-carried devices [9]) and in uncovering statistical laws of such patterns (i.e. inter-contact times distribution [10]) as well as in identifying potential recurrence and predictability [11], which could be used for discovery in opportunistic networks of IoT devices.

The main research problem this thesis addresses is how to acquire knowledge about the availability of devices in the neighbourhood in a distributed fashion in order to optimize the discovery and subsequent communication process in IoT scenarios of opportunistic networking. The aim of this research and thesis is therefore to find learning techniques in order to derive such knowledge with the objective of planning and scheduling the discovery and communication. An ideal smart IoT device should indeed be capable of learning its pattern of interactions with other devices and benefiting from such patterns to schedule its resources efficiently, so that device resources can be used effectively only when such interactions are predicted to occur with a high probability. By finding an algorithm for scheduling resources for such instances when other devices are present in communication range, it is in fact possible to help to prolong device lifetime by avoiding energy waste in devices which will otherwise unnecessarily probe for neighbours when not needed. In addition, in order to provide for a longer communication
time, an ideal smart IoT device should meet the requirements for an optimized discovery with low latency, which tailors communication time according to application requirements. This also means that applications that call for the discovery of a contact at regular time intervals could save energy by avoiding any unnecessary search for contacts in between the contact times. Moreover, by answering the questions of when an opportunistic contact will present itself, and for how long such opportunistic contacts will be present, a node can exploit these contact durations and plan its resource allocation for both discovery and communication. Finally, in order to be as general as possible, such learning algorithms should need to work in IoT devices with low computational power. Therefore, such algorithms must require only a limited amount of data and still operate with a high accuracy. The algorithms must also adapt to different mobility conditions and be aware of how they are performing over time.

1.2 The Challenges

In order to resolve the research problem this thesis poses, several challenges have to be overcome. The following sections introduce such challenges for neighbour discovery which derive from the assumption of IoT scenarios for opportunistic networking. In addition to the previously mentioned benefits introduced by mobility, consisting of increased network connectivity and capacity, reliability and energy efficiency, the major advantage in such settings is that data can be collected, stored and forwarded towards any mobile or static device even when there is no available end-to-end path between the originating node and the sink node. This is accomplished through the mobility provided either by human carriers, or, in some special cases, by controlled vehicles or robots.

Nevertheless, even though from a communication point of view this paradigm introduces several advantages, neighbour node discovery in mobility scenarios is more challenging. Historically, neighbour discovery in static WSNs has been focused on topology formation during long deployment phases. Sensors deployments in the wild usually require a deployment phase which could last several weeks [12]. During such a phase, early-deployed nodes might have to wait for a long time for lately-deployed nodes just to perform a discovery phase that should last few minutes. As a consequence, neighbour discovery in WSN scenarios is aimed at optimizing nodes’ energy expenditure during this starting phase, when the availability of neighbour nodes is unknown. This was initially achieved through the definition of methods for coordinating the temporal overlap of communication between unsynchronized nodes [13]. Traditional discovery implicitly assumes that neighbours are continuously available but subject to an energy-latency trade-off. However, in IoT scenarios of opportunistic networking, this assumption cannot be made, thus introducing new challenges.
1.2.1 Learning and Extracting Knowledge about Mobility Patterns

An important contribution to discovery protocols in IoT scenarios of opportunistic networking is to understand the patterns of encounters between neighbouring devices. In such a way, knowledge about the availability within communication range of either static or mobile IoT devices can be used to introduce additional benefits in the discovery process. With such an ambition, over the last few years, researchers have been trying to analyse the laws of human mobility.

Brockmann et al. [14] have originally studied the circulation of banknotes within the United States of America reporting a distribution of travelled distances which decays as a power law, thus suggesting a Levy-Flight [15] nature for corresponding human walks. Later, thanks to the pervasive diffusion of smartphones, users mobility information collected by network operators has been made available to researchers, thus leading to better insights into individual people's mobility behaviours. Therefore, individual analysis about human mobility has become possible, instead of the collective study obtained through the banknotes diffusion experiment. In fact, Gonzales et al. [9] have found out that users follow exponentially-truncated power law Levy Flights, due to the intuition that their travelled distance mostly points towards a few well-known close locations and only occasionally to farther locations. Rhee et al. [16] have showed that, by exploiting traces with a finer resolution made available by the use of GPS data about human walk patterns, these are susceptible to a truncated Levy-Walk model characterized by heavy-tailed flights (walks) and pause-time distributions. Such a model is also shown to respect a truncated power law for the inter-contact times (defined as the periods of time between subsequent arrivals) survival function, which has been proven by Karagiannis et al. [10] to observe such a dichotomy (power law and exponential tail) over a set of real world mobility traces. Previous works have only been able to provide a hypothesis for such a Complementary Cumulative Distribution Function (CCDF) in the form of either an exponential decay (Grossglauser and Tse [5]) or a power law tail (Chaintreau et al. [17]).

While such works give a first intuition on the statistical laws of human mobility, these conclusions help only to a certain extent to the problem of discovering the availability of nodes in the neighbourhood. Moreover, various mobility patterns could be present, according to the scenario considered. For example, public transportation means such as buses or trains might obey to a more recurrent pattern of encounters, with arrivals which are normally distributed within certain variance ranges. Finally, patterns of controlled mobility might follow an even more strict schedule, which could be found, for example, in applications which make use of drones or robotised mobile data collectors.

One of the biggest challenges and active area of research is to understand which mobility features are needed in order to reach an acceptable level of accuracy and predictability over a wide range of different mobility patterns. Currently, in discovery, the mechanisms which are mostly used in order to adapt the scheduling of resources are based on temporal and spatial
features, as can be seen in Figure 1.1. The simplest way to acquire knowledge about mobility is to understand the evolution of the times at which any node in the network encounters another node (is within communication range) or the time between such contacts, namely, the arrival times or the inter-contact times as well as the durations of such contacts. Under the hypothesis that such features present a certain correlation in time, the nodes could exploit temporal recurrence to predict next arrivals. Similarly, by understanding the number of encounters experienced within a finite time window, nodes can adapt the frequency of their discovery process in order to discover contacts with a lower miss probability. Alternatively, spatial features could be used, such as knowledge about geographical locations, available through the use of Global Positioning System (GPS) receivers as well as knowledge about motion gained through accelerometer sampling or by Received Signal Strength Indication (RSSI) measurements. By understanding locations of nodes it is possible to turn on the radio interface only if another device is predicted to be within communication range with a high probability. Alternatively, by recognizing how far the mobile nodes have travelled or by counting how many nodes are nearby, the probing frequency could be adapted accordingly.

Nevertheless, an important issue and challenge concerning such mobility features, is the way they are obtained. For example, as aforementioned, several approaches based on spatial features require additional hardware (GPS, accelerometers) which not only has a considerable cost and accessibility needs on the IoT devices for its integration, but also needs to be counted in the overall process for its power consumption. For such reasons, approaches that are based on temporal features are usually more interesting, since they only require the capability to
measure time differences on an IoT device. Research on new mobility patterns features to be incorporated into discovery frameworks in order to predict future availability of devices in device neighbourhood is however still required. For example, a promising direction could be to incorporate contextual information about the environment the IoT device is moving into, including knowledge about its preferences, friendship, social behaviour or about the different types of locations it is encountering. In addition, notions about the type of motion it is adopting could be introduced, which can differentiate between mobility patterns while in a public transportation system and while following a human mobility pattern, as well as while using robotised (controlled) data collectors. Moreover, an important challenge is to understand how to efficiently integrate new mechanisms for sharing such knowledge in IoT scenarios, which are inherently distributed in nature. This could lead to an exploitation of such features on a much larger scale. This means that correlation between individual IoT devices’ mobility patterns could be discovered and exploited to introduce smarter approaches, which rely on such knowledge.

Evidently, it is of utmost importance, not only to understand which mobility features explain a detailed history about the IoT devices’ patterns of encounters, but also to capitalize on such knowledge in order to derive accurate predictors able to forecast future interactions, thus exploiting the correlation of such patterns. This means that IoT devices need techniques that are able to learn about such mobility, therefore suggesting the use of machine learning to devise smart approaches in order to acquire such knowledge which could be exploited for predicting future encounters. Some initial approaches, such as the works of Dyo and Mascolo and Shah et al. have already tried to exploit mobility information in order to adjust the neighbour discovery process, but mainly from an energy efficiency point of view. Such frameworks exploit a branch of machine learning, called Reinforcement Learning, used as a means to learn and store information about temporal patterns of encounters, under the hypothesis that IoT devices could exploit such knowledge to adapt their discovery process. While many approaches already exist in the field of data mining, and are capable of predicting mobility patterns, very few have been used in the context of discovery, mainly because they usually require extensive training phases and long data collection phases in order to operate with high accuracy. Instead, reinforcement learning based approaches not only are able to learn online and with very few interactions with the environment, but also require very little computational capabilities, which makes them desirable in resource constrained environments such as in IoT scenarios of opportunistic networking.

Summarizing, the main challenges in learning and extracting knowledge about mobility patterns are:

- How to extract, in an efficient way, knowledge about patterns of encounters between IoT devices, either from temporal/spatial interactions or from additional contextual knowledge and if and how to share information between distributed IoT devices.
1.2 The Challenges

- How to learn such knowledge with the objective to understand and predict when and for how long there will be future encounters between devices with high probability, thus allowing for the optimization of the discovery process.

1.2.2 Resource Scheduling for Optimized, Low Latency and Energy-Efficient Discovery

Assuming the availability of a mobility learner capable of acquiring knowledge about the patterns of encounters between IoT devices, a further challenge for discovery approaches in IoT scenarios for opportunistic networking is the optimization and planning of the scheduling of resources in an energy and latency efficient way (Figure 1.2). Many discovery protocols in research provide for a trade-off between the energy and latency they need while discovering neighbours. By duty cycling (alternating between ON and OFF state) their radio in order to make them sleep for part of their time, they provide a way to trade energy for latency in environments which are fundamentally resource constrained and in which communication is delay tolerant. However, with the introduction of opportunistic networking, IoT devices might be subjected to opportunistic contacts in which time windows for communication are too short for transmitting all the required data, thus requiring high throughput (usually high power) radios. Evidently, under these circumstances, trading latency for energy should only be per-
formed when devices are not in contact in order to provide for an extended lifetime. Conversely, when nodes are in contact, a fast, latency optimized discovery should be provided in order to optimize communication time for contacts that might have a short duration and might be rare and essential to be able to communicate the necessary amount of data in the lifetime of the device.

One of the main challenges for a discovery approach is to recognize other devices’ presence in the neighbourhood within a time window that should be finite and not longer than the experienced contact duration. However, usually, such a duration might vary over time, thus posing a challenge on how to adapt the discovery process in order to correctly recognize every contact. In fact, if the resource schedule is not adapted in such a way, it could mean that a device might not be always recognized, therefore, leading to a $\leq 100\%$ discovery probability. In real world mobility conditions, contacts might be arbitrarily long depending on a plethora of factors, such as IoT devices’ relative speed and motion direction, communication ranges and many others. Therefore, by probing with a predefined fixed rate, some contacts could be missed. Nevertheless, by correctly recognizing these opportunistic contacts, the discovery process contributes to the knowledge acquisition process, ultimately leading to a working learning process where data comes from the environment through interactions which can be used for predicting future contacts. However, without a guaranteed discovery latency, contacts might often be recognized towards the end of their interactions, leaving a very small time window (named residual contact time as shown by Anastasi et al. [22]) for the actual communication between devices. This means that potentially long contacts, might be underutilized for communication purposes. For such reasons, and considering that contacts might be rare, neighbour discovery for opportunistic networking in IoT scenarios might require a dynamic adaptation of the discovery schedule to satisfy application requirements. For example, a requirement could be about the necessary communication time needed, after discovery, to be able to transfer correctly all the data. Therefore, by being able to exploit knowledge about mobility patterns made available by a mobility learner, such a process could be simplified. Predicted arrivals and contact durations could therefore be used in order to help in achieving a more efficient protocol operation.

A different challenge for neighbour discovery arises from the need to recognize not only the presence of devices, but also their absence, with the additional constraint of preserving the energy of IoT devices. By learning and knowing when nodes are not present, energy can be preserved thus extending the lifetime of IoT devices which are resource constrained by nature. Therefore, power management techniques based on mobility patterns are desirable in order to work with IoT devices which usually run on battery. However, IoT devices should still provide the capability to recognize unexpected IoT device encounters, even if they are short. This implicitly sets a limit on the amounts of energy that could be saved on such devices. Consequently, IoT devices shall avoid wasting resources when other devices are predicted to be within communication range with a low probability, thus improving power consumption on both static and mobile nodes with respect to a scenario where such a feature is not considered.
Knowledge about mobility patterns allows for planning the communication along with the discovery process, in order to decide how many resources to dedicate for contacts. For example, an important role is played by the knowledge about the duration and the next forecast arrivals. This allows saving energy by discarding contacts if possible, in lieu of a next forecast contact which might be longer or more significant for the relaying of data. For example, an application might need to exploit a future contact in order to route the packets to the best nodes for relaying the information, avoiding the need to discover meaningless contacts, thus benefiting the overall energy.

Summarizing, the main challenges for resource scheduling aimed towards an optimized, low latency and energy-efficient discovery are:

- How to optimally schedule the resources to be used for neighbour discovery in order to reduce energy wastage when nodes are not in communication range, and, at the same time, allow for exploitation of the maximum contact duration for useful communication time when nodes are within communication range.

- How to plan the discovery and communication based on learned patterns of encounters between IoT devices in order to optimize the resources and identify meaningful contacts worth exploiting.

1.3 Contributions

This thesis examines neighbour discovery in IoT scenarios for opportunistic networking and postulates that current protocols require a knowledge driven approach in order to optimize the entire neighbour discovery process. It asserts that the lack of knowledge about mobility patterns leads to an unoptimized discovery process, where energy is wasted and guaranteed communication time is not provided. By learning and exploiting mobility, it is shown that future contacts between IoT devices can be predicted with a certain degree of accuracy. Further, this thesis explores the optimization of resource scheduling in order to devise low latency and energy efficient discovery protocols. By exploiting the acquired knowledge about mobility, optimization and planning of communication can thus be achieved.

This thesis makes the following contributions:

- The development of a learning based framework for Context Aware Resource Discovery (CARD) in IoT scenarios of opportunistic networking. The framework is capable of learning in which way to schedule more or less energy, in order to adapt the discovery process to the underlying mobility patterns. Optimization of energy expenditure when contacts are not present is provided, as well as a low latency and asynchronous discovery in order to provide for optimized communication time, subject to application requirements in the form of a periodicity parameter.
• An evaluation of the context aware resource discovery framework based on extensive simulations in scenarios of opportunistic networking under different mobility scenarios: controlled periodic patterns of robotised IoT devices (i.e. drones); public transportation systems mobility (i.e. buses) with periodic Gaussian patterns distributed within certain intervals; real world mobility (i.e. human mobility) such as office environment based patterns. A performance evaluation and a comparison with relevant state-of-the-art has been made using metrics such as energy efficiency, latency and discovery success ratio.

• The implementation of an Arrival and Departure Time Prediction (ADTP) algorithm based on Least Squares Temporal Difference (LSTD) learning. Such an algorithm predicts the next arrival and departure times relying only on temporal data, thus not requiring additional hardware components or energy. A generally applicable algorithm has been devised, which works online without requiring any extensive offline data collection and training phases, thus capable of making accurate predictions with only a limited amount of data. The algorithm provides accuracy estimates about the predictions by relying on a short history of the differences between the actual and predicted values. Resiliency mechanisms are incorporated, in order to recognize and act in case of abrupt changes in mobility patterns.

• An evaluation of the accuracy of the time prediction algorithm based on different realistic and synthetic traces. Different mobility patterns (controlled periodic, public transportation systems, real world traces) have been tested in order to evaluate the accuracy and the effects of the settings of the parameters. The resiliency to abrupt mobility pattern changes has been tested in such scenarios, in order to overcome situations in which abrupt variations in periodicity manifest themselves.

• The development of a neighbour discovery framework for IoT scenarios of opportunistic networking, capable of planning and optimizing the discovery process and the subsequent communication process based on knowledge coming from the arrival and departure time prediction algorithm. The framework adopts asynchronous time-slotted and latency-bounded discovery mechanisms, which do not require time synchronization between neighbouring devices and are generally applicable.

• An analysis of the performance of the discovery framework, based on different realistic and synthetic traces. The various aforementioned mobility patterns have been tested in order to evaluate the energy efficiency and the latency achievable with such a framework in comparison with the state-of-the-art.
1.4 Organisation

The subsequent chapters of this thesis present the above-mentioned contributions in detail and are structured as follows:

- Chapter 2 introduces the relevant background necessary to understand the current state-of-the-art of neighbour discovery for opportunistic networking in IoT scenarios. By surveying relevant protocols of mobility agnostic and mobility aware discovery, the work presented in the following chapters is motivated.

- Chapter 3 introduces this thesis’s first contribution for Context Aware Resource Discovery (CARD) in IoT scenarios for opportunistic networking. The proposed system model is reported along with details covering the learning model and the discovery strategy adopted.

- Chapter 4 illustrates this thesis’s Arrival and Departure Time Prediction (ADTP) and Discovery Framework for IoT scenarios of opportunistic networking. The proposed algorithm for predicting arrival and departure times is covered in detail along with the planning framework for optimized discovery actions.

- Chapter 5 covers the implementations of this thesis’s contributions with the objective of performance evaluation. The extensions to a network simulator and to a reinforcement learning framework are reported along with details concerning the implementations of the proposed contributions.

- Chapter 6 covers the simulations details and the performance evaluation results. The performance of the two discovery frameworks with respect to power consumption and latency is reported along with the results on the accuracy of the prediction framework.

- Chapter 7 concludes this thesis with considerations about the contributions with respect to how they tackle the research problem and what has been achieved as well as future research plans.
Chapter 2

Background and Related Work

Neighbour Discovery has been a very productive area of research over the last decade, especially following the introduction of device mobility. This chapter introduces the background literature and gives an exhaustive overview about recent trends in the current state-of-the-art.

Section 2.1 gives a brief overview about neighbour discovery approaches in IoT scenarios of opportunistic networking, by differentiating them according to the way they benefit from the knowledge about IoT devices’ mobility. Section 2.2 briefly overviews mobility agnostic discovery approaches and classifies them according to their need of time synchronization. Section 2.3 introduces to mobility driven discovery protocols which are divided into classes according to which mobility features they exploit. Section 2.4 concludes this literature review with some discussions on discovery and on the focus of this thesis.

2.1 Neighbour Discovery for Opportunistic Networking in IoT scenarios

Neighbour Discovery protocols have been originally introduced as a means to solve power consumption issues at deployment, in static networks of wireless sensors. One of the major research problems at the time was to save energy on resource constrained IoT devices, which needed to form a topology during deployment phases lasting long time windows (i.e. weeks) [12]. Evidently, the naive solution of leaving the radio always awake on such devices would deplete their energy sources in a few hours or days, therefore leading to unsuccessful deployments. For such reasons, algorithms for achieving energy savings with a trade-off on discovery latency were proposed. Such algorithms lead to the duty cycling concept, which describes the percentage of time that the IoT devices’ radio needs to stays awake over a time window. By allowing very low duty cycles a significant amount of energy could therefore be saved on IoT devices by not undermining their energy sources over weeks-long deployments. Moreover, such duty cycling
Background and Related Work

protocols still allow discovering neighbours with high probability within few minutes. However, the sole introduction of very low duty cycles can not solve the neighbour discovery problem in scenarios where topologies are changing over time due to disruption or mobility of devices.

In fact, over the last few years, due to the introduction of mobility, a new communication paradigm has been made possible by the prospect of opportunistically interacting between static and mobile IoT devices. This new Opportunistic Networking [7] concept allows the relaying of data between any pair of devices, even in absence of a predefined end-to-end path between them and consequently introduces new challenges for neighbour discovery. A typical IoT scenario of opportunistic networking, involves mobile IoT devices which typically collect data from statically deployed IoT devices and, based on their encounters, forward such data to other IoT devices. For example, as depicted in Figure 2.1, a man’s mobile IoT device such as a smartphone (B) could collect information from a local area network in an office (A) and forward such data opportunistically through a static IoT device deployed in a newspaper stand (C), which could be collected by a delivery man’s mobile IoT device (D). Furthermore, such data could be delivered to static IoT devices in a building network (E) and collected by another

![Figure 2.1: IoT scenario of Opportunistic Networking.](image-url)
person’s mobile IoT device, which could travel in a taxi (F) and encounter other vehicles, such as buses (G) where other people could sit along with their smartphones. Such people could ultimately relay the message to a static network of IoT devices deployed in a house (H). Evidently, in such a scenario, the task of neighbour discovery assumes the role of finding the patterns of availability of devices in the neighbourhood over time, in order to relay data in the absence of an end-to-end path between devices.

Figure 2.2: Main areas of research in Neighbour Discovery for Opportunistic Networking in IoT scenarios.

It is possible to divide neighbour discovery approaches for opportunistic networking in IoT scenarios into two major classes by differentiating them based on the assumptions they make about the need of mobility knowledge in order to perform discovery. As can be seen in Figure 2.2 it is therefore possible to identify:

- Mobility Agnostic approaches, which do not benefit from the knowledge about mobility patterns in order to find neighbours, but instead rely on time synchronization between devices in order to perform resource scheduling.

- Mobility Driven approaches, which exploit knowledge about patterns of encounters between devices in order to achieve an optimized discovery process, by relying on features of such patterns.

In between mobility agnostic approaches, it is further possible to identify two other classes which are distinguished by the assumptions they make on time synchronization:

- Time Synchronized protocols, which rely on the presence of a common time reference shared across all the devices involved, therefore requiring either connectivity to periodically update such a reference (i.e. with a Network Time Protocol or NTP [23]) or a way
to independently retrieve such information (i.e. GPS receivers [24, 25], ad-hoc synchronization or reference clock compensation techniques).

- **Asynchronous** protocols, which do not rely on any form of synchronization, but instead rely either on the capability of triggering an indirect request for discovery in an IoT device or on the properties of particular sequences of wakeup schedules in order to guarantee an overlap between them within finite time.

Moreover, asynchronous approaches can be divided into two different major classes according to the assumptions they make on the mechanism used to achieve discovery:

- **Indirect Request Driven** protocols which exploit the possibility to trigger an indirect request for wake up without using their primary radio, but instead relying on either secondary lower power radios [26, 27] or customized receiver capable of operating in an RFID-like manner [28], therefore consuming very little energy.

- **Temporal Overlap Driven** protocols that leverage overlapping between wakeup schedules which adopt a slotted model and are based on properties of particular number sequences such as number’s theory and combinatorics properties (i.e. difference sets or Chinese remainder theorem) [29, 30].

Finally, within mobility driven protocols, it is possible to differentiate the discovery approaches based on the features used for acquiring knowledge about mobility patterns:

- **Temporal Knowledge Based** protocols that exploit information such as arrival times, inter-contact times or time of day and duration of contacts [20], as well as rate of arrivals [31] or rush hours [32] in order to adapt the schedule of resources in an optimized fashion.

- **Spatial Knowledge Based** protocols which leverage knowledge about geographical location of IoT devices [33] (i.e. from GPS receivers) or about relative movement and distance between IoT devices (i.e. from accelerometers [34] or signal strength) as well as about co-location [35] of such devices in order to adapt their discovery process.

The next sections introduce to such discovery approaches for opportunistic networking in IoT scenarios.

### 2.2 Mobility Agnostic Discovery Protocols

The first family of neighbour discovery protocols relies mainly on techniques which do not profit from any knowledge about mobility patterns. Such protocols build either on the mechanism with which scheduling of device communication is performed or on the possibility to indirectly recognize other devices’ presence.
2.2.1 Time Synchronized Protocols

Time Synchronized protocols benefit from the availability of a time reference on IoT devices in order to synchronize their temporal schedule for the purpose of discovering each other. For example, as can be seen in Figure 2.3, three IoT devices (A,B,C) adopt the same awake times duration and the same wakeup interval, which is synchronized to a common time reference shared across nodes.

In the ZebraNet experiment [24, 25], IoT devices equipped with a GPS receiver are attached to zebras in order to monitor them. Due to the exploitation of GPS receivers as a means for time synchronization, IoT devices in such a wildlife scenario are able to agree a temporal wakeup slot in which they could discover themselves and communicate, thus avoiding energy wastage but still guaranteeing a latency bound on communication. Keshavarzian et al. [36] show that ad-hoc synchronization protocols can help in defining temporal wakeup patterns for the times at which nodes wakeup along multi-hop network paths. This allows applications in such scenarios to provide for a delay sensitive operation and a fast discovery. Herman et al. [37] report mechanisms for synchronization and discovery between temporal partitioned IoT devices. Such a synchronization is achieved by exploiting either slots overlap mechanism based on relative prime numbers or by either randomly or systematically placing additional slots. Ghidini and Das [38] show that with the aid of synchronization and a Markov Chain they can optimize the discovery process in both energy and latency. Their approach reduces the number of the radio transitions between awake(ON) and sleep(OFF) states as well as reducing discovery latency by allowing a reduced slot length. In fact, such transitions need to be taken into account by protocols since they are not negligible in both time and power consumption.
The Recursive Binary Time Partitioning (RBTP) by Li and Sinha [23] minimizes the discovery latency by adopting a Network Time Protocol (NTP) that allows the synchronization of wake up instances within temporal frames between asymmetric IoT devices (i.e. IoT devices with different duty cycles). WizSync by Hao et al. [39] shows that ZigBee can be used to overhear Wi-Fi beacons as a means to achieve synchronization. Similarly, Camp-Murs and Loureiro [40, 41] present an Energy Efficient Discovery (E²D) Wi-Fi approach, which uses an access point (AP) synchronization mechanism leveraging announcement frames containing timestamps and cluster ID information. Finally, FlashLinQ by Wu et al. [42] exploits a new PHY/MAC layer synchronous architecture operating in a licensed spectrum aimed at improving over previous 802.11 protocols. Such an architecture shows an energy efficient, synchronized, low signal to noise ratio (SNR) communication on a discovery channel which allows finding up to a few thousand devices over a 1 km communication range.

While these time synchronized approaches typically outperform asynchronous protocols in the discovery latency due to their synchronous nature, they however suffer from the complexity deriving from their requirement of having periodic connectivity to maintain synchronization. When such connectivity is not available or devices need to change their resource schedule autonomously (without sharing such knowledge), asynchronous protocols might outperform such synchronous protocols. In addition, in many applications, retrieving a time reference is not always possible due to the lack of hardware (i.e. GPS receivers or real time clocks).

2.2.2 Asynchronous Protocols

Asynchronous protocols do not generally benefit from any kind of synchronization mechanism in order to achieve discovery. They can be divided either into mechanisms that are capable of waking up another radio indirectly or into approaches that rely on a high probability of overlap between awake times of devices which use properties of particular number sequences.

**Indirect Request Driven:**

Indirect Request Driven protocols exploit the capability to wakeup another device indirectly through either a secondary low power radio or a customized receiver, as can be seen in Figure 2.4. In the first case, the secondary radio is typically a ZigBee or Bluetooth low power radio, while the main radio is generally a Wi-Fi high power radio. In the second case, instead, a customized ad-hoc receiver is added in order to trigger the wakeup of the system by relying on the energy contained in the RF signals, as it happens in RFID tags.

The Sparse Topology Energy Management (STEM) by Schurgers et al. [26, 27] introduces a dual radio setup which allows for parallel discovery and communication. This introduces significant power savings thanks to the separate wakeup radio (also called wakeup “plane”) which reduces power consumption under the assumption of sporadic communication events. Wake on Wireless by Shih et al. [43] exploits a secondary low power radio used in combination with a
2.2 Mobility Agnostic Discovery Protocols

Figure 2.4: Indirect Request Driven Discovery.

primary 802.11 radio in order to reduce the power consumption for discovery. This approach shows an improvement of 117% over a single 802.11 radio in power save mode. Geographic Random Forwarding (GeRaF) by Zorzi et al. [44, 45] adopts an approach similar to STEM, but in which the sender is capable of recognizing busy “tones” (i.e. beacons with no information) that are issued by the receiver, thus avoiding collisions. Pipeline Tone Wakeup by Yang and Vaidya [46], similarly to STEM adopts the dual plane radio setup, but has the objective of minimizing the end to end communication delay. In order to achieve such a task, it exploits the plane differentiation, thus allowing to wake the next hop up in advance.

Similarly to Wake on Wireless, Pering et al. [47] analyse the energy, latency and throughput trade-offs obtained by employing different combinations of radio technologies, such as Bluetooth, ZigBee and Wi-Fi. The authors show an improvement in power consumption when a lower power radio is used for waking a higher power radio up indirectly. ZiFi by Zhou et al. [48] exploits the spectrum overlapping between a lower power radio such as ZigBee and a higher power radio such as Wi-Fi. By sampling received signal strength indication (RSSI) measurements, the authors show that Wi-Fi beacons can be recognized with a good accuracy, therefore in a low power mode. Finally, Qin and Zhang [49] report a ZigBee and Wi-Fi dual radio setup, which allows for a parallel Wi-Fi and ZigBee wakeup scheduling. Such scheduling is capable of waking up in advance Wi-Fi through ZigBee when delay requirements for communication need to be met, therefore avoiding to wait for the next scheduled Wi-Fi wakeup.

Radio Triggered wakeup receivers are introduced by Gu and Stankovic [28] as a means for allowing near zero power consumption on IoT devices’ receivers. The authors show that, by using the energy contained in radio frequency (RF) signals, it is possible to wake close range IoT devices up from sleep states indirectly. Such an architecture removes the requirement for duty cycling on the receiver. The capability to convey addressing information in the RF signal at the transmitter is later introduced by Ansari et al. [50] as a means to differentiate senders. In that work, receivers are woken up only if they belong to a particular set, identified by decoding
the received wakeup packet encoded at the transmitter through a Pulse Interval Encoding scheme. In a similar way, Takiguchi et al. \cite{51} use a Bloom filter, which is a probabilistic structure built to test the membership to particular sets. Such a mechanism allows recognizing and differentiating wakeup packets, typically with a low false wakeup probability. Van Der Doorn et al. \cite{52} report a prototype wakeup radio which reduces interference from near GSM bands at 868MHz through the use of a band pass filter and a microcontroller-based, digital filter, therefore reducing the probability of false wakeups due to interferences. Gamm et al. \cite{53} instead modulate at the transmitter a low frequency (125kHz) wakeup signal on the main carrier at high frequency through an On-Off Keying (OOK) modulation. This work and the works by Liang et al. \cite{54} and Wendt and Reindl \cite{55}, benefit from a 125kHz IC at the receiver capable of demodulating the wakeup signal in order to trigger a system wakeup. However, Liang et al. uses it for preamble detection and Wendt and Reindl in a frequency diversity setting.

Several front-end implementations are present in research, with varying power consumption and features. Pletcher et al. \cite{56,57} report a 2GHz customized receiver with Bulk Acoustic Wave (BAW) filter capable of reaching 65\(\mu\)W and 52\(\mu\)W of power consumption. Similarly, Huang et al. \cite{58} show a 51\(\mu\)W receiver which can operate at 915MHz and 2.4GHz frequencies. Le-Huy and Roy \cite{59} show another implementation that further reduces consumption to 20\(\mu\)W and uses a Pulse Width Modulation (PWM) for address comparison. Durante and Mahlknecht \cite{60} present another implementation with reduced power consumption, reaching values of 10\(\mu\)W. RFID-based wakeup radios are used in CargoNet by Malinowski \cite{61} reaching 2.8\(\mu\)W of power consumption. Moreover, Marinkovic and Popovici \cite{62} achieve 270nW of power consumption for Body Area Networks (BAN) applications at 433MHz, while Oller et al. \cite{63} present a sub-1\(\mu\)A receiver by using a Surface Acoustic Wave (SAW) filter, thus having a power consumption of the\(\mu\)W order. A completely passive solution is built by Ba et al. \cite{64} through the combination of a RFID tag and a TelosB node, while Kamalinejad et al. \cite{65} report a solution which harvest its required energy entirely from the wakeup signal.

While these indirect request driven protocols are capable of optimizing radio receivers through customized ad-hoc implementations, they suffer from short range of operation which limits their use to proximity applications or indoor and other close range scenarios. In addition, they require hardware modifications or additional secondary radios, which might not always be available and would require an additional cost for their inclusion in IoT devices.

**Temporal Overlap Driven:**

Temporal Overlap driven protocols rely on the capability to leverage properties of overlapping with high probability between number sequences or randomized intervals. An example can be seen in Figure 2.5 where unsynchronized IoT devices achieve temporal overlapping of awake intervals. The Birthday protocols of McGlynn and Borbash \cite{12} show that, by randomly selecting awake slots in IoT devices, due to the Birthday Paradox, such devices discover each
2.2 Mobility Agnostic Discovery Protocols

The Birthday Paradox \cite{66} states that, by considering an increasing number of people, the probability of finding two of them with the same birthday increases as the number of people increases. Random Asynchronous Wakeup (RAW) by Paruchuri et al. \cite{67} exploits the same principle and achieves discovery by randomizing the awake times of IoT devices in dense scenarios. Balachandran and Kang \cite{68} adopt a similar probabilistic discovery but further add the complexity of the multiple frequencies at which discovery needs to be performed. Their protocol shows an increase in the discovery latency as the number of frequencies increases. Vasudevan et al. \cite{69} compute the discovery time of probabilistic protocols by adopting a Coupon Collector’s problem analogy, showing a $ne(\ln n + c)$ expected time in presence of $n$ neighbours, where $e$ is Euler’s number and $c$ an arbitrary constant. You et al. \cite{70} add over the previous work the possibility for IoT devices to duty cycle, therefore transforming the problem into a $K$ Coupon Collector’s problem, where $K = 3\log_2 n$ and $n$ is the number of neighbours. The work reports a lower and an upper bound on the expected discovery time of $ne\ln n + cn$ and $ne(\log_2 n + (3\log_2 n - 1)\log_2 \log_2 n + c)$, respectively, with $c$ as an arbitrary constant and $e$ as Euler’s number. Vasudevan et al. \cite{71} extend their previous work to a multi-hop communication scenario, reporting a $O(\Delta \ln n)$ running time, where $n$ is the number of neighbours and $\Delta$ is the network’s maximum degree.

Grid quorum based protocols, by Tseng et al. \cite{72}, guarantee a double temporal overlap between awake slots of neighbouring nodes every $n^2$ slots. This is achieved by selecting independently a row and a column of a $n \times n$ matrix of beaconing intervals, which represents the slotted scheduling the nodes have to respect. Jiang et al. \cite{73} extend the quorum protocols to: a $t \times w$ torus quorum (where $tw = n$) \cite{74}, a difference-sets based cyclic quorum \cite{75} and a hypergraph based finite projective plane quorum \cite{76}. Chao et al. \cite{77} report an Adaptive Quorum-based Energy Conserving (AQEC) protocol which changes its grid size according to the traffic load, reducing the grid size in order to discover more neighbours when the traffic is heavier. Zheng et al. \cite{78} introduce methods for designing the optimal blocks of wakeup slots,
which are based on difference-sets from combinatorics theory and guaranteed symmetric (i.e. same duty cycle) discovery within bounded time. Lai et al. [79] extend the grid and the cyclic quorums to asymmetric scenarios, by constructing quorum pairs (two different schedules) and allowing nodes to follow either one of the two schedules in the network. Choi et al. [80] report an adaptive hierarchical approach, based on multiplicative and exponential difference-sets, which is used to provide several levels of power saving and therefore introducing further asymmetry. Similarly, Carrano et al. [81] adopt a nested approach, where superslots are defined in order to deal with asymmetry between nodes’ schedules.

Disco by Dutta and Culler [29] presents a practical way for selecting the duty cycles in discovery protocols as the reciprocal of a prime number $p$ and guaranteeing an overlap within finite discovery latency thanks to the Chinese Remainder Theorem’s congruence property for prime pairs. As we will see in the next chapters and below, Disco is selected as general underlying discovery protocol in this thesis’s contributions, mainly for its practicality of use. U-Connect, by Khandalu et al. [82], improves over Disco in the asymmetric case, by allowing $(p+1)$ awake slots every $p^2$ (hypercycle) slots, in addition to the wakeup every $p$ slots. Searchlight by Bahkt et al. [30] defines a protocol which deterministically searches for overlaps by leveraging fixed anchor slots and moving probe slots within a period. The protocol is also capable of randomizing its probe slots in order to achieve a faster, average case, discovery. McDisc, by Zhang et al. [83], extends such protocols to a multi-channel scenario, by either randomly or deterministically switching between multiple channels in order to search for temporal overlaps.

Jain et al. [84] shows that by imposing the energy burden for discovery on the mobile node (deemed easily rechargeable), a significant amount of energy can be saved on the static node in asynchronous discovery. In addition, Anastasi et al. [22, 85] report about the implications for an asynchronous discovery protocol in scenarios where contacts are short and nodes are moving. Yang et al. [86] introduce an optimal schedule for asynchronous discovery with respect to energy and latency, based on transmission, sleep and listening scheduling. Similarly, Zhou et al. [87] show that, under power law distributed contact durations, if the schedules of IoT devices respect the rule of $T_{ON} \geq T_{OFF}$ and $\tau \geq 2(T_{OFF})$, where $\tau$ is the minimum contact duration, such devices can guarantee an energy saving of $\min\{0.5, \frac{\tau}{T}\}$, where $T = T_{ON} + T_{OFF}$ is the duty cycle period. Trullols-Cruces et al. [88] reach similar conclusions by analysing trade-offs of power consumption with miss probability. Finally, Feng and Li [89] report an analysis of the trade-offs between nodes miss probability and probing frequency combined with their transmission range, showing that, as the frequency and range increase, the miss probability decreases.

While temporal overlap driven protocols might present some limitations in the achievable latency when applied to Bluetooth or Wi-Fi technologies, they do not need any form of synchronization for guaranteeing overlap of awake times between neighbouring nodes. This makes them more generally applicable in comparison to either time synchronized or indirect driven protocols. For such reasons, Disco has been selected as the underlying discovery protocol over
which this thesis’s frameworks for discovery are built. In particular, Disco has been selected for its “practical” approach to achieve discovery which relies on prime numbers overlap between awake slots. It is important to note that, any other temporal overlap discovery protocol such as Searchlight or U-Connect could have been used instead, without compromising the operativeness of this thesis’s proposed contributions.

2.3 Mobility Driven Discovery Protocols

Mobility driven discovery protocols rely on knowledge about IoT devices’ mobility patterns which is used to understand when encounters are likely to occur with a higher probability. Such protocols allow organizing the schedule of the resources in an energy efficient way, avoiding to waste energy when devices are present with a low probability and adapting to changes in the environment due to node mobility.

2.3.1 Temporal Knowledge Based

The approaches relying on temporal knowledge of IoT devices’ mobility patterns show that, by acquiring knowledge about temporal metrics concerning encounter patterns, an optimized discovery approach can be obtained. For example, as can be seen in Figure 2.6 arrival times or rate of encounters knowledge can be used to adapt the discovery process.

Chakrabarti et al. [90] exploit the predictable mobility of public transportation systems in order to learn about the IoT devices’ presence in a startup phase. In a secondary steady phase, the authors exploit such learned knowledge in order to introduce additional power savings in the network. Similarly, Jun et al. [91] introduce a power management framework based on previously collected knowledge about statistics of contacts duration and waiting times between contacts. Dyo and Mascolo [19] use reinforcement learning to adapt their beaconing frequency in a temporal slot based on the encounter frequency of the same temporal slot of the previous
Background and Related Work

day and on an energy budget. Jun et al. [92] adopt a multiple radio approach based on combined low and high power radios and on contact arrival rates and bandwidth, which are used to estimate wake up intervals. The Resource Aware Data Accumulation (RADA) by Shah et al. [20] uses Q-Learning to learn how to schedule duty cycles based on inter-contact times and time of day at which contacts were made. Due to its learning capabilities, RADA is selected as state-of-the-art benchmarking approach, as justified in the discussion below. Sensor Node Initiated Probing for Rush Hours (SNIP-RH) by Wu et al. [32] uses knowledge about the rush hours in a day in order to schedule more resources when the average contact duration is higher. Kondepu et al. [93] combine Q-Learning with an interleaved long or short range beaconing (the result of previous works [93, 94]) in order to learn when to schedule a higher duty cycle for receiving short range beacons, whereas otherwise scheduling a lower duty cycle. Gao and Li [95] define a wakeup scheduling mechanism based on the prediction of future node contacts, by relying on a stochastic modelling of the contact process. Similarly, Zhang et al. [96] model a power law distribution of inter-contact times in order to predict the optimal arrival and departure times to wakeup and save energy in between.

Drula et al. [97] report a mechanism for dynamically adapting the Bluetooth protocol parameters according to the recent contact arrival rate, by increasing the probing frequency when contacts are more likely to arrive based on such history. Similarly, Choi et al. [98] show an Adaptive Exponential Beacon (AEB) protocol, which exponentially relaxes the probing intervals as fewer contacts are detected. Kam and Schurgers [99] extend the previous work by exploiting local information (i.e. mobility, packet queues and expiration times and battery conditions), generally made available by routing protocols, in order to introduce further optimization in the discovery process. Wang et al. [100] introduce a short term arrival rate estimation protocol, which uses previous time slot and time of day information in order to estimate next arrival rates. The eDiscovery by Han and Srinivasan [31], similarly to previous works, increases the beaconing interval when peers are discovered, whereas otherwise resetting it to its minimum value. Zhou et al. [101] exploit temporal contacts history in order to compute the expected values of the number of encounters to be arriving on a per slot basis. Finally, Wi-Fi Sensing with aGing (WiSaG) by Jeong et al. [102], similarly to previous works, relaxes or increases the sensing interval according to the aging property of the inter-contacts distribution, which is the time that has passed since the last contact.

Temporal knowledge based protocols exploit statistical knowledge about times and frequency at which contacts occur in IoT scenarios of opportunistic networking. Historical information is typically used to derive heuristics capable of adapting the probing times in order to reduce power consumption, but very few protocols actually learn about mobility patterns (i.e. RADA). In this thesis, RADA is selected as state-of-the-art reference since the objective of this thesis is to derive techniques for acquiring knowledge about mobility patterns in order to exploit it for planning the discovery process. It is this author’s opinion that learning techniques can better adapt to different mobility conditions (i.e. controlled mobility, public transportation
systems based mobility and human mobility) and provide for low latency and energy efficient discovery protocols.

2.3.2 Spatial Knowledge Based

The approaches relying on spatial knowledge of IoT devices' mobility patterns show that, by relying on knowledge about devices’ positions and their movement, as well as knowledge about their co-location, it is possible to optimize the discovery process. For example, as shown in Figure 2.7 IoT devices use their knowledge about movement or about co-location in order to schedule and adapt wakeups.

The Connection-less Sensor-Based Tracking System Using Witnesses (CenWits) by Huang et al. [103] adopts a scheduling mechanism for the probing frequency in search and rescue applications which depends on the speed of the mobile IoT devices. The speed is used by the authors for deciding how often to schedule wakeup times for the IoT devices of hikers. Banerjee et al. [104] introduce throwboxes for Delay Tolerant Networks (DTN), which are static IoT devices equipped with dual radios. The throwboxes exploit location, speed and direction information contained in beacons of mobile IoT devices, captured by long range radios, as a means to wake up in advance low range high throughput radios if a contact is predicted. Bread-crumb by Nicholson and Noble [33] leverages location information combined with throughput information in order to forecast connectivity availability.

Similarly, Blue-Fi by Ananthanarayanan and Stoica [105] predicts the availability of Wi-Fi connectivity by combining Bluetooth contact patterns with cellular tower location information as well as with received signal strength (RSSI) based movement knowledge. Footprint by Wu et al. [106] also uses movement knowledge obtained by observing cellular towers ID and RSSI measurements in order to trigger Wi-Fi access point scans only if an IoT device has moved enough to cause a change of context. Sivaramakrishnan et al. [107] report an algorithm for sampling the displacements of moving IoT devices. By relying on accelerometers measures and
on an Artificial Neural Network (ANN), such an algorithm learns and predicts the distribution of IoT devices, adapting the discovery. Li et al. [108] exploit an autoregressive model based on location and direction history in order to compute and share with its neighbours their mobility estimate (in order to correct their estimate), which is used to adapt the frequency of discovery. WiFisense by Kim et al. [109] reports an algorithm for deriving the optimal Wi-Fi scanning interval which employs mobility movement knowledge retrieved by sampling accelerometers, as well as access point density and average RSSI measures. The Mobility Assisted User Contact (MAUC) Detection by Hu et al. [34] leverages accelerometer sampling in order to trigger Bluetooth scans only when users are classified as moving, by adjusting the Bluetooth scans according to an exponential increase, multiplicative decrease backoff technique. PISTONS [110] and PISTONSv2 [111] use the notion of speed in order to adapt the discovery process. However, while the first version uses a predefined maximum speed, the second version assumes nodes can estimate their mobility.

Borbash et al. [112] report an algorithm which uses probabilistic slotted discovery in combination with knowledge about the number of neighbours in order to maximize discovery. The Context Aware Power Management (CAPM) by Xi et al. [113] exploits the sharing of wakeup schedules between neighbours in order to optimize power consumption. Tumar et al. [114] expand such an algorithm towards multiple radio based discovery, where a low power radio is combined with a high power radio for discovery. Luo and Guo [116] leverage the properties of Code Division Multiple Access (CDMA) with the objective of multi user detection in discovery. Similarly, Zhang and Wu [117] detect when a flocking condition occurs in order to increase probing frequency for adapting to a crowded environment. WiFlock by Purohit et al. [118] defines a protocol which coordinates and synchronizes listening and communication intervals when a flock condition occurs in order to allow for group formation. NetDetect by Iyer et al. [119] adapts the beaconing rate of IoT devices by using an estimate of the neighbour density. Such a distributed algorithm, has the property of converging the transmission probabilities towards the optimal values.

Karowski et al. [120] define optimization techniques for (slotted) listening intervals and durations, as well as for switching between channels in a multi-channel scenario. The Cooperative Duty Cycling (CDC) by Yang et al. [121] shows that, when a clustering condition occurs, significant power savings can be introduced in a flock by cooperatively lowering their duty cycles. United we find, by Bakht et al. [122], exploits a dual radio setup, where high range, high power radios are used in order to reach distant IoT devices not reachable by low range low power radios, which are instead used to save power when communicating ranges are short. Finally, in Acc by Zhang et al. [35], a framework for accelerating slotted discovery in dense scenarios based on shared wakeup schedules between nodes is presented.

Spatial knowledge based protocols exploit co-location and knowledge about movement, geographical location and distance between devices in order to adapt their discovery protocols. However, many of these approaches require additional hardware (i.e. GPS receivers or ac-
celerometers) therefore, increasing both the energy and the cost of the IoT devices in which they are used. Finally, very few approaches actually learn and predict about mobility, or combine temporal and spatial knowledge in order to introduce additional efficiency in discovery.

2.4 Shortcomings and Discussion

Time synchronized discovery protocols require a common time reference, which needs to be refreshed periodically in IoT devices in order to make them maintain a coherent value and operate correctly in the discovery process. Various reference sources are used in pursuit of such a synchronization objective. IoT devices usually synchronize their clocks with either ad-hoc techniques or network time protocols. However, such methods often require frequent connectivity between nodes in order to disseminate the reference information. In scenarios of opportunistic networking frequent connectivity cannot always be guaranteed, therefore possibly affecting the reliability of these protocols. Nevertheless, when such connectivity is present, synchronized protocols can outperform asynchronous protocols, especially in the capability to guarantee a latency optimized discovery with low power operation. However, if IoT devices want to autonomously change their schedules, they still require coordination in order to adapt to meet other node’s needs. Moreover, most of the approaches neglect an accurate analysis on the incurring energy cost of adding a synchronization protocol. Some approaches also need additional hardware in order to derive such a time reference. For example, few approaches exploit GPS receivers and real time clocks, which are usually either required by the application (i.e. location monitoring [24, 25]) or are present by hardware design (i.e. smartphones [23]). This means that, whenever such additional hardware is present, the additional energy cost needs to be taken into account. On the contrary, when such hardware is not present or it is not easily integrable, or even cost too much for the application, time synchronized protocols cannot be used.

Indirect request driven protocols require the availability of an additional piece of hardware, such as either a wakeup radio or a secondary lower power radio. While in some works the use of such customized ad-hoc radios introduces a quasi-negligible power consumption at the receiver, unfortunately, it carries along the limitation of such radios in communication range and therefore limits their use mainly to indoor or other close range scenarios. Moreover, most of such approaches do not consider optimization on the sender radio, and sometimes even modify it to consume more energy than it would need for a standard communication task in order to achieve few more meters of communication range. Such an inadequacy in communication range becomes even more important in IoT scenarios of opportunistic networking because shorter ranges translate directly into shorter contact durations between IoT devices. A better approach is followed by those protocols that benefit from combination of high throughput, higher power radios with lower power but higher range radios. In fact, even though in IoT scenarios of opportunistic networking contacts might be relatively short and scarce, by exploiting a
lower power but higher range radio, the higher power radio can be woken in advance to exploit
the entire contact duration. This allows increasing the useful communication time and
network capacity without incurring the higher power consumption of higher throughput radios
for discovery, therefore with a lower power discovery.

Temporal overlap driven protocols do not typically need any synchronization and are deemed
the most generally applicable protocols due to their capability to work without requiring any
particular additional piece of hardware. However, while such approaches have been shown to
work in ZigBee nodes, their applicability to other radio technologies such as Bluetooth and Wi-
Fi has some limitations. In fact, in time slotted protocols, radios are required to have fast and
very frequent turn on and turn offs, as well as short awake times in order to achieve a very low
latency and a correct operation of such protocols. An issue with Wi-Fi is pointed out by Bakht
et al. in Searchlight [30], where the required setup time for waking up Wi-Fi from user space
reaches 1 second, therefore limiting such a discovery protocol. Furthermore, in Bluetooth, the
recommended default value for the inquiry duration is of 10.24s, which can be lowered to 5.12s
as proven by [123]. However, such temporal values are several orders of magnitude higher than
ZigBee’s values. For example, in WiFlock, slot durations have been shown to achieve 80µs,
which is the time necessary on a standard IoT device to perform a Clear Channel Assessment
(CCA). Moreover, in IoT scenarios of opportunistic networking, short contacts require IoT
devices to meet latency requirements in order to comply with application requirements. While
many approaches guarantee the possibility to bound latency (i.e. Disco, U-Connect), protocols
of probabilistic nature (i.e. Birthday protocols) cannot guarantee such a limit. However, such
approaches typically achieve a lower average latency in comparison to the latency bounded
protocols. A tighter integration between both approaches (for example as shown in Searchlight)
could therefore show better performance overall, specifically, on average and in the worst case.

Temporal knowledge based protocols belong to the class of mobility driven protocols, which
exploit temporal knowledge about mobility patterns in order to provide for an optimized discov-
ery approach. These protocols exploit knowledge about arrival times or frequency of encounters
in the form of a collective metric (i.e. rush hour, recent activity level) or history of encounters
collected over time (arrival rate, inter-contact times) in order to optimize the discovery pro-
cess. However, many of the works that exploit such a temporal knowledge might risk having
a significant number of failures in node discovery, which are due to the statistical nature of
these proposed algorithms. Therefore, these temporal knowledge based works do not offer a
high level of accuracy in the process of adapting the probing times and frequency according
to the mobility patterns. In addition to their need for improvements in the accuracy, such
works fail to guarantee a bound on the discovery latency. This means that such works fall
short in assuring that the discovery process is capable of providing enough time for communi-
cation, in IoT scenarios of opportunistic networking where contacts might be short. In fact, if
a neighbour is discovered towards the end of an interaction because of a very low duty cycle, a
significant amount of the available time for exchange of data between devices becomes wasted.
None of the approaches shows the capability to meet application requirements in the useful time needed for communication, subject to the mobility patterns. This means that applications cannot set and adjust the discovery process according to their needs and save resources by autonomously adjusting their resource scheduling. Moreover, no actual planning of the communication is performed, which could allow deciding whether to discover for contacts or discard meaningless short contacts based on metrics such as contact duration, depending on application requirements. Finally, very few of the protocols adopt a discovery protocol that is capable of operating in different mobility conditions, such as in periodic controlled mobility (i.e. drones and robotised data collector), periodic with a Gaussian distribution of inter-arrivals (i.e. buses or trains) or in real world human mobility. This means that, if the mobility conditions change, the discovery protocols require adaptation of the parameters in order to adapt to a change in the mobility patterns.

Spatial knowledge based protocols differ from the temporal knowledge based approaches in the source of knowledge about mobility patterns they use. They typically require the availability of knowledge about geographical location, co-location, movement (acceleration) or distance between devices. This means, that, in contrast to the temporal knowledge based approaches, they need additional hardware capable of offering such type of information, such as GPS receivers or accelerometers. This hardware however requires energy and an additional cost for its inclusion, if the application does not need it and therefore needs to be added. For example, in location monitoring applications or in smartphones, GPS receivers are largely used. Therefore, in such settings, location knowledge would come free of the additional energy and inclusion cost. A few works actually explore ways for reducing power consumption on such hardware (i.e. Paek et al. [124] or Liu et al. [125]), which could be used in combination with these discovery approaches. In addition, many approaches for predicting trajectories and location based on mining of large datasets are reported (see Lin and Hsu for a survey [126]), but very few actually try to exploit online learning approaches, which do not require training or “big” datasets to operate. Moreover, while many works exploit co-location between nodes in order to share their schedules to adapt to dense scenarios, very few of the works presented actually try to share the schedules between nodes which are not in contact but are supposed to be in contact in future in order to coordinate multiple nodes discovery. Finally, alternative hardware could be exploited in order to derive new sources of knowledge. For example, acoustic or luminosity sensors could be used to infer people’s presence in determined condition and or to gain a better insight on the context in which nodes are moving or are deployed.

2.4.1 Contribution to State-of-the-art

The focus of this thesis is on deriving temporal knowledge based approaches, mainly because they offer a more general approach than spatial knowledge based protocols. In fact, temporal knowledge based approaches do not require costly additional hardware or additional power
consumption to derive such spatial knowledge. In addition, this thesis identifies in the asynchrono-
ous temporal overlap driven protocols the most generally applicable mobility agnostic
approaches. Our contributions build on top of such general purpose approaches by adding
a mobility driven component that helps to derive optimized discovery protocols. Temporal
overlap driven protocols, in fact, do not require any form of synchronization between IoT de-
vices, which can independently set duty cycles and still discover each other. Moreover, some
recent works provide for a guarantee in latency which can be exploited to build application
requirement aware discovery protocols.

New discovery protocols have therefore been proposed in this thesis, in order to bridge
the gap in the current state-of-the-art. Very few temporal knowledge based protocols [20, 19,
107] are in fact capable of learning about mobility patterns, in particular about the temporal
sequence of arrivals an IoT device might experience. This shows the need for discovery protocols
that are aware and adaptive to the temporal features of patterns of encounters, as these are
learned over time. Many protocols try to reduce power consumption by adapting to the contact
pattern, but very few actually try to predict [95, 33] when the nodes will be present in order
to turn off the radio when nodes are isolated from any neighbour. Moreover, no protocols
try to increase and guarantee a minimum communication time either per contact, or overall,
as an application requirement. In fact, many sensory applications require data collection on
a periodic basis and, if such applications call for the discovery of at least one contact every
predefined time period, they could save energy in between, avoiding discovering unnecessary
and unmeaningful contacts. Furthermore, an algorithm should not only learn and adapt to
the mobility pattern, but also predict when and for how long a contact in the future will
occur. By incorporating knowledge about time windows in which contacts will appear, IoT
devices can plan the scheduling of the communication. For example, an application dependent
planner could allocate resources for communication in a much more efficient way by knowing
such an information, with respect to a greedy scheduler. Moreover, very few algorithms are
capable of learning online, therefore without any form of training. A good knowledge based
approach, in fact, should be capable of avoiding offline data collection phases and training
phases, especially when such phases depend on the data given. An ideal algorithm should
therefore adapt to different mobility conditions (i.e. periodic, public transportation systems
based, human mobility based) and incorporate mechanisms to recognize changes in the mobility
patterns in order to adapt to such changes. Finally, none of the current approaches presents
a way to provide for accuracy estimates about the predictions in order to define a way for
discovery protocols to modify their resource schedules according to how good or how bad they
perform.

It is therefore possible to summarize the main advancements that this thesis covers with
respect to the state-of-the-art as follows:

- The introduction of temporal knowledge based methods which do not require neither ad-
  ditional costly hardware (i.e. GPS or accelerometers) nor additional power consumption.
• The definition of frameworks for learning mobility patterns that build on underlying highly applicable asynchronous discovery approaches, which can be used on a wide range of devices.

• The definition of optimized resource schedulers which are capable of leveraging knowledge about encounter patterns to introduce optimization in energy expenditure and procedural latency in the discovery process.

• The possibility to forecast when and for how long a contact will occur in order to plan the discovery and the communication phase.

• The definition of mechanisms for guaranteeing latency in discovery, which applications can exploit to satisfy requirements to provide a minimum communication time period.

• The definition of learning based approaches which can adapt to different mobility conditions and recognize changes in mobility patterns in order to adapt to changing scenarios.

• The introduction of an online learning system which is able to work with very few data, requiring little computation and that is able to produce accuracy estimates on its performance.
Chapter 3

Context Aware Resource Discovery

In this chapter, the Context Aware Resource Discovery (CARD) approach for IoT scenarios of Opportunistic Networking is presented. After an introduction about how CARD helps in contributing towards this thesis’s research problem, an analysis and a discussion of relevant state-of-the-art discovery protocols, as well as an introduction to relevant learning frameworks and algorithms is presented. Furthermore, the proposed system model for context aware resource discovery is reported in a separate section along with considerations about IoT scenarios of opportunistic networking. Moreover, the context aware learning model is described in detail and discussed in all of its parameters and configurations, as well as in its integration with the discovery protocol. Finally, the chapter is concluded with considerations on how CARD helps closing the gap with respect to other state-of-the-art approaches.

3.1 Introduction

CARD helps in contributing towards solving this thesis’s research problem by building a learning model based on a Reinforcement Learning algorithm from the Temporal Difference methods, named Q-Learning, which is briefly introduced in Section 3.2.2. CARD’s learning model is able to learn the patterns of encounters between IoT devices by acquiring knowledge about the sequential temporal allocation of contacts over time.

CARD provides for optimization of the discovery process through a trial and error learning procedure which schedules resources aimed at discovering IoT devices with a low latency when contacts are learned to be present with a high probability. In addition, CARD avoids energy wastage by reducing the power consumption and scheduling less resources when contacts are learned to be present with a low probability. Moreover, due to its particular schedule definition,
it allows for an additional “selective” sleeping of the radio for part of the schedule based on discovery results, which allows for a further reduction of power consumption, as it will be shown in detail in the following sections. In fact, as we will show also in such an evaluation section, current state-of-the-art solutions provide for low power consumption, but cannot optimize it, as CARD provides even lower consumption levels combined with additional communication time provision.

One of the most important advantages of CARD is the possibility to tailor the discovery process to application requirements. In fact, discovery frameworks should be able to be customized in an effortless manner in order to be personalized according to the needs of an IoT application. Such needs include the capability to provide for the necessary communication time subject to availability of communication opportunities, as well as, the possibility to avoid energy wastage in resource constrained IoT devices. In CARD, as it is shown in the next sections, applications might decide, based on their communication time requirements how to provide for a certain latency (hence a certain communication time) by scheduling less or more intense probing actions.

Finally, in order to provide for a framework for wide usability in a heterogeneous IoT device environment, CARD adopts both reinforcement learning and the broadly applicable asynchronous discovery protocols previously described in Section 2.2.2. In fact, CARD’s learning framework needs a very low computational power and requires no training, as well as offers the capability to be applied to varying mobility patterns. Moreover, the asynchronous discovery protocols, and in particular the latency-bounded temporal overlap driven protocols, require no time reference and synchronization, thus being generally applicable to a wide range of IoT devices. However, since CARD uses time slotted temporal overlap driven protocols as their underlying discovery approaches, as pointed out in Section 2.4, this could introduce limitations in the granularity of the worst case latency bound when used in combination with Bluetooth or Wi-Fi radios.

### 3.2 Current discovery approaches analysis and discussion

Temporal overlap driven asynchronous protocols are the most generally applicable mobility agnostic protocols, therefore the most interesting ones for IoT scenarios of opportunistic networking, where devices can be heterogeneous and present different features. Among those protocols, several approaches are capable of providing a practical and application customizable discovery by letting few parameters decide the duty cycle and the maximum latency to expect according to that schedule. Such latency bounded discovery protocols are preferable in opportunistic networking scenarios where short and rare contacts need to be recognized within a certain latency window, in order to exploit the residual contact time for useful communication. Examples of such protocols are Disco by Dutta and Culler [29], U-Connect by Khandalu et al. [82] and Searchlight by Bahkt et al. [30].
Temporal knowledge based protocols are the most generally applicable mobility driven protocols, due to the fact that they do not require any additional hardware in order to derive such knowledge. In between temporal knowledge based approaches, it is possible to identify the learning based techniques as the most interesting ones since they allow acquiring and storing knowledge about different and various patterns of encounters IoT device might experience, as well as predicting IoT device returns. This allows to have a general approach which will work regardless of the mobility pattern such IoT devices are subject to (i.e. periodic, public transportation system based or human mobility based). RADA by Shah et al. [20] and the work by Dyo and Mascolo [19] consider Reinforcement Learning [21] as a preferable paradigm, due to its low computational complexity which makes it largely applicable to heterogeneous IoT devices. In addition, by operating in a trial-and-error way, it better models an online learning process which learns over time and does not need any training or any a priori data collection phase.

In this section, a brief analysis of Disco is presented, since in this work and simulations it is used as the underlying discovery protocol in combination with the governing context aware approach that adapts the use of resources according to the mobility pattern. However, any other protocol with the same features as aforementioned can be used as a baseline protocol. Moreover, a short introduction to reinforcement learning is reported, as beneficial for this thesis’s reader, especially with focus on Q-Learning [127], as it is the algorithm exploited by both RADA and CARD. Finally, an analysis of the most recent learning based protocol, RADA is also presented in order to identify its shortcomings. This protocol is also used in the evaluation section in order to use it as a benchmark comparison with CARD.

### 3.2.1 Disco

Disco adopts a practical slotted discovery model, which makes it very easy to be used by CARD’s scheduler. Latency bounds and duty cycles can be easily and autonomously computed by every IoT device independently based on a few parameters, as it will be shown in this section. However, any other protocol with the same features (i.e. U-Connect or Searchlight) could be used as a substitute, since for CARD’s purposes only a latency bound on discovery is required. Slotted discovery models such as Disco, divide time into slots of fixed duration, known to all the IoT devices, in which the devices’ radio can be either awake listening or transmitting (or performing a combination of both) or sleeping. In the simplified version of the algorithm, any two IoT devices $i$ and $j$ need to select two numbers $m_i$ and $m_j$ that are relatively prime (coprimes). These numbers represent the number of slots after which every IoT device needs to wake up for one slot. The generic $k$-th IoT device therefore sleeps for $m_k - 1$ slots and wakes up at the $m_k$-th slot, with a duty cycle equal to the reciprocal of such an interval: $d = \frac{1}{m_k}$. By adopting such a constraint for the schedule, the Chinese Remainder Theorem [128] guarantees that there is an overlapping slot every $m = m_im_j$ slots.

For example, by considering two IoT devices with $m_i = 5$ and $m_j = 2$, it is possible to
obtain the situation depicted in Figure 3.1, where it can be clearly seen that there is an overlap every $m = m_i m_j = 5 \times 2 = 10$ slots. In such a situation, even if the IoT devices start counting their slots at different times ($t = 0$ for node $i$ and $t = 1$ for node $j$) there is a periodic overlap at $t = 5 + 10k$, with $k \in \mathbb{Z}_+$ and 10 is the aforementioned $m$ constant, which is a function of the schedule of both IoT devices.

![Figure 3.1: Disco between IoT devices $i$ and $j$ with coprime pair $m_i = 5$ and $m_j = 2$.](image)

Since by considering this simplified version of the algorithm would cause problems when IoT devices might want to select their duty cycles autonomously, the authors have proposed a dual prime number approach. In fact, by autonomously selecting the duty cycles, the numbers selected might not be a coprime pair or could even be the same number on both devices, which would not lead to a successful discovery. In addition, only a handful of numbers usually respect both the duty cycle requirements and the coprime pair rule. In the dual prime number approach, each $i$-th IoT device selects two prime numbers $p_{i1} \neq p_{i2}$ which results in a duty cycle of $d = \frac{1}{p_{i1}} + \frac{1}{p_{i2}}$. Such schedule guarantees a successful overlap between any two IoT devices $i$ and $j$, since at least one pair in the set $\{(p_{i1}, p_{j1}), (p_{i1}, p_{j2}), (p_{i2}, p_{j1}), (p_{i2}, p_{j2})\}$ will be composed of relatively prime numbers. However, as the authors point out, not every choice of prime numbers influences the latency experienced in the discovery between IoT devices in the same way.

The authors distinguish between several factors that can influence the latency based on the selected intra-node (same IoT device) or inter-node (different IoT devices) prime pairs. A first possibility for IoT devices, is to select balanced primes, meaning that intra-node primes are approximately equal to each other or very close, according to the flexibility of the primes choice. For example, this means that for a desired duty cycle of $\simeq 5\%$ the primes pair would be $(37, 43)$. A second choice is to select unbalanced primes, meaning that intra-node primes are significantly different. This means that, in order to obtain the same duty cycle aforementioned of $\simeq 5\%$ the primes pair would be $(23, 157)$. In addition, since by considering prime pairs IoT devices can independently schedule any value, it is possible to distinguish between symmetric and asymmetric pairs. In the case of symmetric pairs, both IoT devices can select the same pair of primes, meaning that inter-node pairs are equal on both devices. For example, both IoT devices could select the prime pair $(23, 157)$. Alternatively, when inter-node pairs are different, IoT devices present asymmetric pairs. For example, IoT device $i$ might schedule $(23, 157)$ while IoT device $j$ might schedule $(37, 43)$. 
Starting from these considerations, the authors have shown that largely unbalanced primes lead to significantly low latency in asymmetric pairs. However, unbalanced primes in symmetric pairs show the highest latency. Moreover, balanced primes in symmetric pairs generally show a good average latency. This means that, whenever possible, as a good policy, IoT devices should select asymmetric and unbalanced primes. The authors therefore propose, that, according to the desired duty cycle, prime pairs should be selected as to have a first prime close to the reciprocal of the envisioned duty cycle and the second prime as a much larger number.

Another interesting property of Disco, is the capability to still guarantee discovery between devices in the presence of misalignment between slots, a condition very likely to occur in real world applications. In fact, Disco transmits a beacon at the beginning and at the end of a slot, thus guaranteeing that, when slots are misaligned, beacons are successfully received. In case slots are perfectly aligned, however, this will cause a collision, which is handled by the application in the way it prefers.

One of the major shortcomings of Disco is its lack of a mechanism for adapting its schedule to mobility patterns. While applications benefiting from Disco could in fact autonomously set their duty cycles according to device mobility patterns, the authors do not provide such a mechanism. However, since Disco in its dual prime pairs version allows IoT devices to autonomously set their duty cycles and primes, and since the authors provide a few formulas to design a latency driven discovery, a context aware mechanism such as the one provided in this thesis could exploit such a feature. For example, in an IoT scenario of opportunistic networking, contacts between devices might be short and need to be recognized within a fixed time window, as dictated by an application requirement on minimal communication time. For the discovery to occur within a latency bound $t_{\text{bound}}$, the following inequality (see [29]) should be in place:

$$p_i \cdot p_j \cdot t_{\text{slot}} \leq t_{\text{bound}},$$

(3.1)

where $(p_i, p_j)$ is the inter-node prime pair between node $i$ and $j$ which leads to the discovery, and $t_{\text{slot}}$ is the slot duration shared across all nodes. This translates into the requirement for the product of primes in the two IoT devices to be:

$$p_i \cdot p_j \leq \frac{t_{\text{bound}}}{t_{\text{slot}}},$$

(3.2)

which, in case of symmetric prime pairs, becomes:

$$p \leq \sqrt{\frac{t_{\text{bound}}}{t_{\text{slot}}}}.$$  

(3.3)

Finally, if balanced prime pairs are considered, the required minimum duty cycle becomes:

$$d_{\text{min}} \geq \frac{1}{p} + \frac{1}{p} = \frac{2}{p} = 2 \sqrt{\frac{t_{\text{slot}}}{t_{\text{bound}}}}.$$  

(3.4)
By exploiting these equations, an application (in this thesis, CARD) can therefore easily define its schedule according to the latency bound needed.

### 3.2.2 Reinforcement Learning and Q-Learning

Reinforcement Learning is a form of machine learning that crosses the boundaries between Supervised and Unsupervised Learning techniques. Due to its foundation on the “Pavlovian” Classical Conditioning theory, the learning occurs thanks to positive or negative reinforcements in response to particular actions an agent decides to perform, which are supposed to influence the behaviour of an agent over time. More specifically, Reinforcement Learning is based on a Markov Decision Process (MDP) which models the evolution over states of a particular environment.

Such MDPs can be considered augmented Markov Chains, composed of states \( s \in S \), actions \( a \in A \), transition probabilities \( P(s, a) \), rewards \( R(s, a) \) and a discount factor \( \gamma \). More precisely, a learning agent in a certain environment could transition between states according to different transition probabilities and the actions taken, as well as the discounted reward it gains by performing a certain sequence of actions-states over time. The objective of learning is therefore to understand how to make sequential decisions in order to solve a problem in which there is limited feedback from the environment. A learning agent will therefore control its action according to some optimal behaviour, usually driven by the discounted sum of rewards over time an agent will receive.

In many practical cases an optimal model of the environment is available, meaning that transition probabilities and rewards are known. When such a model is available, Dynamic Programming techniques can be used to solve the MDP, with the objective to find a policy \( \pi : S \rightarrow A \) that maps states to actions which maximizes the discounted sum of rewards over time. In order to decide which policy is better, a value function \( V(s) \) is constructed to help in the decision process. Therefore, Bellman’s equations (see [129] for more details) for the optimal Value Function and the optimal policy are constructed and can be solved by either Policy Iteration or Value Iteration techniques.

When a model of the environment is not available due to unknown rewards and transition probabilities, Temporal Difference methods (see [129]) are used to solve the learning problem, and for this reason, such methods are also called model-free learning methods. Temporal Difference methods are characterized by a main trait, which is to adjust the value of a particular state based on the immediate reward and the estimated (discounted) value of the next state. This means that, the learning process is a step-by-step process in which at every iteration the agent interacts with the environment and updates the value function online, hence these methods are also called online learning methods. An important task to be performed by such agents is to allow the agent for some exploration of the environment, rather than spending all its time on exploitation of its current optimal policy. In fact, the policy an agent follows
could be suboptimal and some exploration might help the agent in exploring every action and every state of the environment. Different exploration/exploitation trade-offs can be considered, but usually, the \( \epsilon \)-greedy strategy is considered, which consists of randomly selecting between exploration an \( \epsilon \)% of the time and exploitation the remaining \((100 - \epsilon)\)% of the time.

If state-action rewards and value functions (defined as \( Q(s, a) \)) are considered, two temporal difference methods can be considered, namely SARSA and Q-Learning. Q-Learning differs from SARSA mainly in the fact that it is an off-policy learning algorithm, meaning that the agent learns even if the policy followed is not the optimal one. In fact, SARSA is an on-policy learning algorithm, meaning that, at every step, instead of selecting the best state-action value, it will select the on-policy state-action value, thus learning by following the agent’s policy. This solution, however still leads to the optimal policy if all states-actions are tried over time. It is possible to see the original Q-Learning algorithm from Watkins and Dayan \[127\] in Algorithm 1. At the beginning of an episode, defined as an entire trajectory into the state-action space up until the goal state, state-action values are initialized arbitrarily (i.e. to zero or to small random values) and the starting state \( s \) is initialized. At every step, the agent selects an action \( a \) according to an exploration/exploitation policy and states-actions Q values. The action \( a \) is performed and the resulting state is observed, along with the rewards for taking that particular action \( a \). The algorithm then updates its state-action values backwards based on the reward and the difference between the future (discounted) state-action value, if the best action is selected, and also the previous state-action value. Two parameters also influence such a learning process, namely the learning rate \( \alpha \) and the discount factor \( \gamma \). The first parameter \( \alpha \) influences how fast the agent is learning, i.e. how much the agent values new information towards its accumulated knowledge. The second parameter \( \gamma \), instead influences the cumulative sum of future rewards, i.e. how much the agent values new future rewards compared to more immediate rewards.

**Algorithm 1: Q-Learning - Watkins and Dayan (1992)**

```
1 Initialize state-action values \( Q(s, a) \) arbitrarily;
2 repeat
3     Initialize state \( s \);
4     repeat
5         Choose action \( a \) from state \( s \) using policy derived from \( Q \) (e.g. \( \epsilon \)-greedy);
6         Take action \( a \), observe reward \( r \) and next state \( s' \);
7         \[ Q(s, a) := Q(s, a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)] \]
8         \( s := s' \);
9     until state \( s \) is terminal;
10 until:
```
3.2.3 RADA

The Resource Aware Data Accumulation (RADA) framework by Shah et al. [20] provides for an energy efficient algorithm based on Q-Learning that learns the arrival patterns of mobile IoT devices in order to avoid wasting resources when no IoT devices are present. In RADA, the states are defined as a triple \((i_{ct}, i_r, t_{od})\) which represents the inter-contact time, the in range boolean value (either 0 or 1 if nodes are discovered) and the time of day as measured by the sensor node, respectively. In addition, in order to avoid having a very large state space which would not only compromise memory requirements, but also convergence time, the authors propose a Hamming distance [130] based state reduction technique which calculates how much the states are similar in order to merge them. The Hamming distance between any two states \(s_i\) and \(s_j\) is computed according to the following formula:

\[
H(s_i - s_j) = W_1 \cdot |V_1(s_i) - V_1(s_j)| + \ldots + W_k \cdot |V_k(s_i) - V_k(s_j)|,
\]

(3.5)

where the \(V_k\) are state values, \(W_k\) are weights for balancing similarity between states and, in RADA’s case, \(k = 3\) according to state definition. If any two states have a Hamming distance lower than a particular threshold, which the authors set to \(\theta = 1.0\), the states are merged and only one of the two states is considered: at the moment of creating a newer state, an already present state is used instead. The weights \(W_k\) that are used to decide the similarity between states are instead changed according to the mobility considered. This however means that if the scenario in which the IoT device is operating changes, a proper tuning of the values is required.

The actions are modelled as three duty cycling actions, namely, a high duty cycle \((\delta_{max})\), a low duty cycle \((0.5 \cdot \delta_{max})\) and a very low duty cycle \((0.1 \cdot \delta_{max})\). The high duty cycle action is ideally to be scheduled when a mobile IoT device is within range, while the other should be scheduled when the mobile IoT device is out of range with a high probability. In order to define an appropriate action duration, the authors introduced time domains, whose durations represent for how long duty cycling actions are scheduled. The authors also proposed an automatic tuning procedure for such time domains durations, according to 5% of the inter-contact times, which minimizes energy consumption. However, if the inter-contact times are shorter or contact durations are larger, as it might happen in human mobility scenarios, such a strategy might not be the best to adopt. Moreover, in order to provide for a balance between exploration and exploitation, the authors propose an adaptive \(\epsilon\)-greedy strategy with the following formula:

\[
\epsilon = \epsilon_{min} + max(0, (\epsilon_{max} - \epsilon_{min}) \cdot (\epsilon_{max} - \epsilon)/\epsilon_{max}),
\]

(3.6)

which reduces the exploration when a sufficient number of contacts are found in the beginning, therefore serving the purpose only of allowing more exploration at the startup of the learning process. In particular, at the beginning the \(\epsilon\) threshold is maximum \((\epsilon_{max})\) and it is progres-
3.3 Proposed System Model for Context Aware Discovery

Proposed System Model for Context Aware Discovery by Pozza et al. [131] assumes that, in IoT scenarios of opportunistic networking (as described in Section 2.1), static and mobile IoT devices are free to be deployed anywhere or move around and opportunistically enter within communication range of other IoT devices. A typical situation that the proposed neighbour discovery approach solves is the one depicted in Figure 3.2. In such a scenario, deployed infrastructure interacts with multiple mobile IoT devices.

Finally, the reward function is based on the energy spent and on the number of contacts discovered during a particular action. More specifically:

\[ r = (n_c \cdot m_p - 1) \cdot e_s, \]

where \( n_c \) is the number of contacts discovered, \( e_s \) is the energy spent during the action and \( m_p \) is a constant named multiplier price. As more contacts are discovered, a positive reward proportional to the energy spent will be assigned, whereas, if no contacts are discovered, a negative reward proportional to the energy spent will be assigned.
devices moving around according to a certain mobility pattern. It is assumed that every node adopts CARD’s strategy for discovery independently and in a distributed fashion, therefore, allowing every node to autonomously learn the pattern of its encounters. For example, as depicted in Figure 3.2, two mobile nodes X and Y travel independently at different speeds and in opposite directions, encountering several static IoT devices along their way. Node X, will therefore encounter, in order, A, B, C, D, E, F and then again, all of them, starting from A. Node Y, instead, will travel in the opposite direction finding, in order, F, E, D, C, B, A and then again, all of them, starting from F. It is also assumed, for simplicity, that mobile IoT devices will not exploit their contacts, which will be shorter due to the difference in the relative speed of the two moving devices. In addition, it is assumed that close range static IoT devices will discard their eventual contacts, as they do not move anywhere.

In Figure 3.3 it is possible to see the temporal parameters which will be discussed in this and in the following sections. A contact between any two IoT devices is defined as the situation in which both IoT devices are within the communication range of each other. Here contact duration is defined as the time window between the beginning and the end of a contact, as observed by an IoT device. Moreover, an inter-contact time is defined as the time that lasts between the starting times of two subsequent contacts, as seen by the IoT device. For example, in Figure 3.3 it is possible to see the contact durations and inter-contact times that node X experiences along its moving path.

A last metric called periodicity is also introduced, which represents the minimum inter-contact time a node expect to experience. This means that, an IoT device with a certain periodicity is expecting to see, in the time that lasts between the end of the previous contact and the periodicity, up to a maximum of one IoT device (therefore either one or no IoT devices at all). In the next section, it will be shown that CARD uses such a parameter in order to estimate the duration of its actions. This implies that the actions that will be scheduled will have the objective to find up to one contact, if present. Evidently, many parameters affect the inter-contact times experienced over time, such as the speed of the mobile IoT device, the route and direction of approach, as well as the position of the static IoT devices. For example
in Figure 3.2, if the IoT device X moves with constant speed, it will be in contact with all the IoT devices for a fixed contact duration, but experience several different inter-contact times depending on the distance between the IoT devices, e.g. 1200, 1500, 900, 1300, 1500, 1700 seconds. This means that the periodicity parameter will be set to the minimum value, that is 900 seconds in this example. This means, that, as long as the conditions do not change, every 900 seconds the mobile IoT device X will find either one contact or none.

As this periodicity is just a parameter for CARD’s learning process, a possibility is to record for a certain time period the inter-contact times experienced by the IoT device. For example, at the beginning of the learning phase, the IoT device could perform a high frequency probing in order to record the periodicity parameter. As an alternative, the IoT device could keep a small history about the last inter-contact times experienced (i.e. last 10 values) and change accordingly the periodicity parameter as the minimum of these values. Nevertheless, another possibility is to indicate the value for the periodicity directly into CARD’s framework, if there exists any a priori knowledge about the mobility patterns of the particular environment where the node is going to be deployed. Note that this could be also thought of an application requirement, calling for finding at least one contact within a periodicity, if such a contact is present, subject to mobility pattern’s availability. This means that, if more than one contact is experienced (i.e. two contacts in a periodicity), the application might not require the additional contact and discard it, thus saving energy with respect to a solution that instead searches for every contact. However, if the inter-contact time experienced is shorter than the periodicity, since it is assumed to find up to one contact per periodicity, this could mean that some contacts would be missed and there will be more than one contact in one periodicity. Nevertheless, from the point of view of the application, this could be acceptable if it requires just one contact every predefined time period.

3.4 Learning Model for Context Aware Discovery

The main objective of CARD, is to derive a learning model capable of understanding the temporal evolution of a contact pattern. In CARD, an agent learns about the device context, intended as temporal knowledge about encounter patterns, in order to control the schedule of an IoT device, based on such information. This means that contextual knowledge concerning the temporal evolution of mobility patterns (in the form of a contact arrival pattern within a periodicity) is acquired and stored in order to select the best actions over time. Such an approach, relieves the burden of acquiring spatial information on the IoT device, by relying only on the contact patterns. This means that no additional hardware and associated cost, as well as power consumption cost is added by CARD.

CARD aims to adapt to patterns of encounters by leveraging existing asynchronous discovery protocols as baseline approaches. In addition, CARD aims at introducing a learner and a scheduler in order to reduce energy wastage when contacts are not expected, while, at the same
time, trying to exploit the entire contact duration for useful communication. CARD is in fact based on a Q-Learning algorithm which is modelled with the objective of jointly optimizing the power consumption and the latency with which contacts are discovered. However, different from RADA’s learner, which has the objective of optimizing power consumption, the objective of CARD’s learning agent is to find the policy which guarantees a minimal latency when discovering a contact, subject to the learned mobility pattern.

Another advantage of CARD is that, different from RADA, it does not require fine tuning of the parameters according to the different mobility conditions the IoT device is experiencing. In particular, RADA requires a Hamming distance based state space reduction technique in order to not only avoid the explosion of the number of states and its consequent convergence issues, but also to customize the learning context to the mobility pattern considered. In CARD, instead, the context is modelled only according to the periodicity and the contact duration, as it will be seen further in this section, which, under the reasonable assumption of a low ratio between contact duration and periodicity avoids the state space to grow problematically.

The Q-Learning actions of CARD are modelled as a nested slotted sequence of low latency but higher energy discovery sub-actions combined with lower energy but higher latency discovery sub-actions. The framework adopts a learning model, which could be used with any latency-bounded asynchronous temporal overlap driven discovery protocol, such as Quorum based protocols, Disco, U-Connect, or Searchlight. As it will be seen also in this thesis’s implementation and evaluation sections, Disco has been used as a baseline discovery approach for its practicality.

### 3.4.1 Disco-based Schedule Model

The generic Disco action used in CARD, is driven by a latency constraint for the discovery. A balanced primes strategy is adopted, in which, following a given latency bound, the system dynamically builds the schedule considered. While the unbalanced primes strategy would give a better latency, the balanced primes strategy is considered mainly because it provides for good average latency and does not have a performance decline in symmetric pairs (unbalanced primes would show the worst latency in symmetric pairs). A Disco action is therefore defined by its discovery bound and by a slot time, which it is considered known and shared across nodes. Given those two parameters, through Equation 3.3, the algorithm computes a prime “candidate” value \( p \) by considering the equivalence in the inequality. The algorithm then builds the Sieve of Atkin \[132\] sequence of primes up until the candidate value \( p \) and selects the last two values as the balanced prime pair that follow Equation 3.2. For example, by considering a latency bound \( t_{\text{bound}} = 10s \) and a slot time of \( t_{\text{slot}} = 10ms \), it follows:

\[
p \leq \sqrt{\frac{10}{0.01}} = \sqrt{1000} \approx 31.63. \tag{3.8}
\]
By building the Sieve of Atkin up to 31, the last two primes which will be used by the algorithm are (29,31). This will translate into an effective latency bound between two IoT devices, which is:

\[ t_{\text{bound}} = p_i \cdot p_j \cdot t_{\text{slot}} = 29 \cdot 31 \cdot 0.01 = 8.99s \leq 10s. \]  \hspace{1cm} (3.9)

### 3.4.2 CARD Actions Model

Leveraging these asynchronous discovery protocols, the actions for CARD are defined as a slotted and customized sequence of two particular types of Disco actions, namely:

- **Low Latency sub-actions (LLSA)**, which are scheduled for a time \( t_{\text{LLSA}} \) and that guarantee the discovery of a peer performing the same type of action within a low latency bounded time \( t_{\text{low}} \).

- **High Latency sub-actions (HLSA)**, which are scheduled for a time \( t_{\text{HLSA}} \) and that guarantee the discovery of a peer performing the same type of action within a high latency bounded time \( t_{\text{high}} \gg t_{\text{low}} \).

CARD uses such sub-actions as a basic building block of its Q-Learning actions, by composing them of LLSA and HLSA in a particular slotted schedule. A CARD action is defined by the number of sub-actions it is composed of, denoted by \( N_S \), and by a couple \( A(N_{\text{HLSA}}, N_{\text{LLSA}}) \). Such values represent, respectively:

- \( N_{\text{HLSA}} \): the number of initial slots in which the actions schedule high latency sub-actions.

- \( N_{\text{LLSA}} \): the number of slots after the first initial \( N_{\text{HLSA}} \) slots in which the actions schedule low latency sub-actions.

Note, that, from a general point of view, it might be possible that \( N_{\text{HLSA}} + N_{\text{LLSA}} \leq N_S \). In such a case, the remaining \( N_S - (N_{\text{HLSA}} + N_{\text{LLSA}}) \) slots after the initial \( N_{\text{HLSA}} \) and the central \( N_{\text{LLSA}} \) ones, are considered high latency sub-actions. Since considering all the possible combinations of sub-actions would have increased the action space affecting the convergence time, the action space has been reduced to a limited set of actions. In fact, since it is assumed that only up to one contact for action should be found, the number of low latency sub-actions is thus reduced to just one, but presenting itself at different indexes within the action. The action space is therefore reduced to a set of cardinality \( N_S + 1 \) as follows:

- one action composed by \( N_S \) high latency sub-actions denoted as \( A(0,0) \),

- \( N_S \) actions \( A(0,1), A(1,1), \ldots, A(N_S-1,1) \) composed by one low latency sub-action but placed in all the \( N_S \) different positions.

For example, as depicted in Figure 3.4, a \( A(4,1) \) action with \( N_{\text{HLSA}} = 4, N_{\text{LLSA}} = 1 \) and \( N_S = 6 \) for CARD is composed by four high latency sub-actions, one low latency sub-action
and another high latency sub-action. The $A(0,0)$ action is intended for use when no contacts are expected within the action duration, in order to save energy. The other actions, instead, are designed with the intention of mapping the expected contact within the action duration with a low latency sub-action, while trying to save as much energy as possible by mapping high latency sub-actions when the contact is not expected.

### 3.4.3 CARD States Model

CARD’s state definition is based on the pattern of the beacon reception within an action. In practical terms, the states represent an index within an action which indicates in which sub-action the beacon which lead to the discovery was received. More formally, a state is defined as a couple $S(A_D, C_D)$, where:

- $A_D$ represents the absence duration, which is the number of initial sub-actions in which no beacons are received; therefore a number ranging from 0 to $N_S$.

- $C_D$ represents the contact duration, which is the number of sub-actions in which beacons are instead received; therefore, either 0 or 1.

After every action execution, the agent will transition between states based on the eventual beacon reception and will consequently learn which of the actions better maps the states. For example, after the scheduling of the $A(4,1)$ action aforementioned, the agent might recognize the contact through a beacon reception in the second sub-action of the overall six sub-actions. This will lead the agent from the previous state to the $S(1,1)$ state as shown if Figure 3.5.

![Figure 3.5: CARD $S(1,1)$ state reached after the $A(4,1)$ action with $N_S = 6$.](image)

While in RADA, the state definition requires the IoT device to measure the time of day or the inter-contact time, in CARD, only the beacon reception pattern within an action needs to be identified. In addition, as aforementioned, in RADA the state space can explode due to the high number of possible states and requires a Hamming distance state space reduction technique.
which needs to set weights differently according to the mobility patterns. On the contrary, in CARD, the states definition does not need to be changed according to the experienced mobility pattern, but follows directly from the definition of the states and actions.

3.4.4 CARD Actions Schedule Parameters

In addition to the defined periodicity as the expected minimum inter-contact time, another parameter is needed to set the duration of the sub-actions, which is the expected minimum contact duration. As for the periodicity parameter, its value could be either learned online at the beginning or adjusted according to a moving history. However, if an a priori knowledge about the mobility patterns exists or if the application can set a requirement, the minimum contact duration value could be specified in such a way. This means that, if a mobile IoT device is known to move with a speed up to a predefined value due to its mobility conditions (i.e. human carried or vehicle carried), and that it interacts mainly with static IoT devices, by also knowing the radio range in meters, the contact duration could be approximated. For example, by considering a human carried IoT device moving at 3.6Km/h speed, corresponding to a person walking, if the radio range is around 100 meters, a contact of up to 200 seconds could be expected, while such a contact could be reduced to 18 seconds if the IoT device is moving at 40Km/h (i.e. a bus).

Nevertheless, such a value could be decided by the application, which typically requires a minimum contact duration which needs to be discovered such as, for example, discovering all the contacts lasting more than 200 seconds. This means that contacts shorter than 200 seconds will not be guaranteed to be discovered with a 100% probability, and therefore could be either discovered or not. If such a condition occurs, this also implies that the learning process might not receive an accurate feedback from the environment after scheduling its actions, meaning that the underlying state might not be perceived correctly: i.e. there was a contact during a sub-action, but it was not correctly recognized.

Starting from these considerations and, different from RADA which has an automatic tuning procedure for the action durations, the duration of the actions for CARD is defined as follows:

\[ T_A = \frac{P}{T_A} \]

(3.10)

where \( P \) is the periodicity parameter, therefore having an action which lasts for exactly one periodicity. The duration of sub-actions \( t_{LLSA}, t_{HLSA} \), instead, is designed to be close to the minimum contact duration \( D \). However, assigning directly \( t_{LLSA} = t_{HLSA} = D \), thus having the sub-actions which last for a time equal to the minimum contact duration might not be possible, since, only in a few lucky cases an action which lasts a periodicity might be divisible into equally sized sub-actions lasting for the exact minimum contact duration. Therefore, a rounding procedure was performed based on the number of sub-actions in an action. In
particular, firstly the number of sub-actions $N_S$ is computed by:

$$N_S = \lfloor \frac{P}{D} \rfloor. \tag{3.11}$$

The $N_S$ value thus computed is then used to define the sub-actions durations as:

$$t_{LLSA} = t_{HLSA} = \frac{P}{N_S}. \tag{3.12}$$

Here, the floor function of Equation 3.11 gives a safer bound by making it possible that the actual times are greater than the minimum contact duration $t_{LLSA} = t_{HLSA} \geq D$. Moreover, the latency bounds for the sub-actions are set as:

- the low latency bound for the LLSA, $t_{\text{low}}$ is set as 5% of the contact duration $D$, therefore $t_{\text{low}} = 0.05 \cdot D$,

- the high latency bound for the HLSA, $t_{\text{high}}$ is set as 100% of the contact duration $D$, therefore $t_{\text{high}} = D$.

Note that, such a definition for the bounds allows to be sure that, with a low latency sub-action, ideally 95% (in the ideal condition of no errors in the communication) of the contact time should be left after discovery. Similarly, with a high latency sub-action, the bound allows to safely be aware of the contact discovery, thus always having a feedback from the environment. Following such a definition for the actions and states, it becomes evident that the state space and the action space cannot grow indefinitely as in RADA. In fact, the action and the state space have cardinality equal to $N_S + 1$, meaning that they are known once Equation 3.11 is computed. Therefore, as long as the ratio between the periodicity and the minimum contact duration is not too high, the state and action spaces are limited and the convergence is fast.

### 3.4.5 CARD Reward Function Model

The reward function of CARD is modelled to force the agent towards the optimization objective, which is a low latency and energy efficient discovery. Different from RADA in which only power consumption is optimized, in CARD the objective is in fact to drive the scheduling of the actions in order to have a low latency sub-action when a contact is expected with high probability, based on the learned pattern. This means that the encountered IoT device will be found in a faster way and more communication time will be provided for applications to exchange data. Moreover, CARD optimizes also energy consumption as it will try to schedule high latency and low energy sub-actions when a contact is expected with low probability. Based on the assumption that up to one contact will be present within a certain action’s scheduled duration, CARD’s learning agent will try to schedule the actions in the aforementioned way. This means that, over time, different actions will be tried with the objective of learning the
sequence of actions that maximize the discounted cumulative sum of rewards. As the agent tries
different actions, different states will be reached based on the mobility patterns. The agent,
thanks to the specific design of the reward function, will learn over time which is the best
sequential decision for scheduling actions in order to match the mobility pattern. The optimal
policy which approaches the mobility patterns will be learned and, at every step, the agent
will approach the contact with the best action which maximizes the reward.

In CARD, the reward function is based on the action $a$ and the state $s'$ reached by following
such an action. It is therefore assumed that the beacon reception pattern following an action
decides the reward. This also means that it is considered that every sub-action scheduled will
be able to identify the presence or the absence of a contact within its scheduled time, which
is reasonable by assuming that the contact is longer than the worst case latency bound for
the high latency sub-action. Therefore, under the assumption that every scheduled action will
return a correct feedback from the environment, the reward function is defined as follows:

$$R(s, a, s') = R(a, s') = \sum_{i=1}^{N_S} B_i \cdot C_i,$$

where $B_i$ is the sub-action beacon reception constant and $C_i$ is the sub-action cost. The beacon
reception constant $B_i$ is assigned as follows:

$$B_i = \begin{cases} 
+1 : & \text{beacon received during } i\text{-th sub-action} \\
-1 : & \text{beacon missed during } i\text{-th sub-action}
\end{cases}.$$

(3.14)

The sub-action cost $C_i$ is assigned as follows:

$$C_i = \begin{cases} 
1 : & \text{high latency sub-action} \\
N_S - 1 : & \text{low latency sub-action}
\end{cases}.$$

(3.15)

Evidently, such a reward definition allows for the following consequences:

- a beacon received during a low latency sub-action will have a positive and higher reward
  than a beacon received during a high latency sub-action,
- a beacon missed during a high latency sub-action will have a negative but higher reward
  than a beacon missed during a high latency sub-action.

This means that, if an action leads to a miss in every sub-action and therefore to the state
$S(N_S, 0)$, the action that will have the highest negative reward $-N_S$ would be the $A(0,0)$
action, because it is composed of only high latency sub-actions. Every other action would in
fact lead to a lower reward of:

$$-1 \cdot (N_S - 1) + (-1) \cdot 1 + \ldots + (-1) \cdot 1 = -2 \cdot N_S + 2,$$

(3.16)
because they are composed of one low latency sub-action and $N_S - 1$ high latency sub-actions. When an action instead leads to a discovery in any of its sub-actions, the reward has three possibilities, according to the fact that the action matches the discovery with a low or high latency sub-action. In the first and optimal case, a beacon is received during the low latency sub-action and during the other sub-actions the agent performs high latency sub-actions, leading to an overall reward of 0:

$$+1 \cdot (N_S - 1) + (-1) \cdot 1 + \ldots + (-1) \cdot 1 = 0.$$ (3.17)

For example, in Figure 3.6, the CARD state $S(1, 1)$ is reached after the $A(1, 1)$ action with a reward of $+5 - 1 - 1 - 1 - 1 - 1 = 0$. In the second case in which the action recognizes the beacon with high latency by scheduling an action with a low latency sub-action, the reward becomes as follows:

$$-1 \cdot (N_S - 1) + (-1) \cdot 1 + \ldots + (-1) \cdot 1 + 1 = -2 \cdot N_S + 4.$$ (3.18)

This situation is depicted in Figure 3.7 where the CARD state $S(1, 1)$ is reached after the $A(4, 1)$ action with a reward of $-5 - 1 - 1 - 1 + 1 = -8$. Finally, in the third situation in which a contact is recognized with an action composed by all high latency sub-actions
reward is computed as follows:

\[
\sum_{i=1}^{(N_S-1)} (-1)^i + 1 = -N_S + 2. \tag{3.19}
\]

Summarizing, the rewards given in this way allow for guiding the system to have higher reward for saving energy out of the contact and jointly discover with low latency when a contact is learned to be expected.

### 3.4.6 CARD Learning and Additional Parameters

Different from RADA, where an initially higher exploration strategy is allowed, a constant 5\% $\epsilon$-greedy exploration strategy is selected. This allows the system to continuously try to explore the environment in order to find more rewarding policies. The Q-learning learning rate $\alpha$ is set to a high value of 0.9 to allow for a fast and always reactive learning. Moreover, the discount factor $\gamma$ is set to a low value of 0.1 in order to make the agent more myopic in order to value more immediate rewards rather than long term rewards. Finally, a selective sleeping strategy is introduced as a further way to reduce power consumption. Such a strategy allows for sleeping as soon as the discovery is performed until the next action starts. For example, if a contact is found in the first sub-action of an action with $N_S = 6$, this means that potentially 5 sub-action durations will be used for saving power consumption on the radio by completely sleeping. However, in the worst case of finding a contact in the latest sub-action, no power consumption can be saved. Nevertheless, an application that requires up to one contact during a periodicity can safely discover such a contact, communicate and then completely turn off its radio for a time period up until a new action needs to be scheduled, without interfering with the learning and discovery process.

### 3.5 Conclusions

In this section, a Context Aware Resource Discovery (CARD) platform is presented, which helps in closing the gap in literature by allowing the learning of mobility patterns between IoT devices in a latency efficient way. Different from the current state-of-the-art solutions, which only show low power consumption, CARD further optimizes the power consumption between IoT devices by adopting a selective sleeping strategy which turns the radio off completely once a contact is found. In addition, by scheduling low latency actions only when IoT devices are learned to be present within communication range with a high probability, combined with a planning of the scheduling of high latency actions, which schedules them when the contacts are expected with low probability, CARD can further reduce power consumption and provide more useful communication time as compared to what is provided by the current state-of-the-art solutions.
This guarantees a discovery process which is jointly optimized in latency and energy and which is capable of providing a guaranteed communication time, subject to mobility patterns and tailored to application requirements. Current state-of-the-art, indeed, does not provide the possibility to tailor the discovery process according to application requirements.

Moreover, different from current state-of-the-art, CARD does not require manual adjustment of the parameters according to the mobility patterns it needs to model, but instead it adopts a general model for the learning states. This allows the model to work in any mobility condition, being it a periodic pattern such as in controlled robots (i.e. drones) or a more normally distributed periodic pattern such as in public transportation systems (i.e. bus) as well as being a human mobility pattern.

Furthermore, CARD has no requirements to measure the exact time of day or inter-contact times in which contacts occur, but just exploits the beacon reception sequence pattern. Moreover, CARD’s state and action space dimensions are completely defined by their parameters, which means that they cannot grow indefinitely as in state-of-the-art solutions, but are limited, thus improving the speed of convergence of CARD’s learning algorithm.

Finally, thanks to the adoption of reinforcement learning algorithms, which requires low computational power, no training or previous model about the environment, and due to the use of asynchronous, temporal overlap driven and latency bounded discovery protocols, this protocol can be generally applicable to any IoT device, thus covering the vast heterogeneity of the IoT world.

Summarizing, it is possible to identify CARD’s benefits in:

- The possibility to build on generally applicable temporal overlap driven asynchronous discovery protocols which can be used on a high variety of sensing devices.
- The definition of a general algorithm which does not require additional hardware but is capable to learn in a trial-and-error fashion just by exploiting the beacon reception pattern within a fixed time window.
- The introduction of an algorithm capable of learning how to adapt to patterns of encounters in order to optimize power consumption of sensing devices.
- The possibility to adapt to an application’s communication time requirement which results in a latency optimized approach allowing to exploit contacts in their entirety.
- The definition of a general model for the learning behaviour, which can adapt to different mobility conditions.
- The introduction of a learning approach requiring only a limited volume of data, no training and very little computational capabilities.
Chapter 4

Arrival and Departure Time Prediction and Discovery

In this chapter, an Arrival and Departure Time Prediction (ADTP) and Discovery Framework for IoT scenarios of Opportunistic Networking is presented. After an introductory section on how ADTP helps in solving the research problem this thesis engages with, an overview of the advanced temporal difference learning algorithm that ADTP benefits from, is presented. Furthermore, the proposed algorithm for the prediction of time of arrivals and departures is discussed in detail, covering all of its configuration parameters. Moreover, a planning strategy for energy efficient and low latency discovery that is based on this prediction framework for next contacts, which improves over state-of-the-art solutions, is reported. Finally, the chapter is concluded with considerations on ADTP’s contributions with respect to state-of-the-art approaches.

4.1 Introduction

ADTP’s contribution towards this thesis’s research problem is twofold. The first contribution as compared to state-of-the-art solutions is in the possibility to predict when and for how long an opportunistic contact will manifest itself in future based on learned patterns of encounters. The second contribution follows from the knowledge acquired about mobility patterns, which allows IoT devices to plan their resource allocation for both discovery and communication.

ADTP is in fact able to predict not only the next arrival times and the next departure times, but also the next contact durations as the difference between such values. The optimization in ADTP is not based on a trial-and-error action scheduling, as in CARD, but instead is based on an actual prediction about the contact arrival and departure times. This allows tailoring the discovery process with low latency when there will be an opportunity for communication.
with high probability and, on the contrary, save as much energy as possible when arrivals are predicted with low probability. This is achieved with a high latency probing which is scheduled in conditions of low probability of arrivals, combined with a selective sleeping feature which tries to further reduce the power consumption when the predictions are accurate, thus sleeping instead of doing a high latency probing.

ADTP therefore allows tailoring discovery by concentrating more resources within reasonable intervals around the predicted contacts, whose durations are computed as a difference between the predicted departures and the predicted arrivals. Resources are thus scheduled dynamically for a time window which depends on the predicted contact duration. Moreover, out of such a predicted time window, resources are saved in order to prolong device lifetime.

ADTP provides a generally applicable framework, which can be used in heterogeneous IoT device scenarios. IoT devices might in fact be constrained in energy resources, with low computational power and without any time synchronization. The adoption of reinforcement learning techniques and asynchronous discovery protocols is therefore justified by these constraints present in IoT scenarios. Reinforcement learning in fact requires no training and a limited volume of data to operate, as well as low computational power. In addition, temporal overlap based asynchronous discovery protocols provide latency guarantees and require no synchronization, thus they are generally applicable to any IoT device.

The learning model of ADTP exploits only temporal knowledge about mobility patterns without requiring additional hardware for acquiring spatial information (i.e. accelerometers or GPS receivers) about mobility. In addition, ADTP provides a mechanism for recognizing abrupt changes in mobility patterns and for adapting the learning process in varying mobility conditions. Finally, accuracy estimates are also provided as a means to estimate the performance of the predictor over time and to coordinate the resource allocation for optimizing the discovery process.

4.2 Learning Algorithms Based on Temporal Differences Methods

The current state-of-the-art approaches for discovery, such as RADA and CARD, use Q-Learning as their algorithm. In fact, Q-Learning allows for the optimization of a policy an agent follows over time, driven by a reward function. This means that the objective of reinforcement learning in such problems, is learning how to control an agent to follow an optimal sequence of actions which maximizes the reward over time. However, as Sutton shows in his pioneering work [133], the temporal difference methods have been initially conceived as prediction methods. Effectively, this means that an agent uses its learned past experience to predict about the future behaviour of an agent in an environment. A classical example for the temporal difference learning is the step-by-step refinement of the prediction of a variable, i.e. the weather
prediction for Sunday refined from Monday through Saturday. After each day, at every step, an agent refines its prediction about the weather on Sunday, based on the feedback it receives from the environment.

After a study of learning frameworks, it has been understood that mobility patterns could better fit a prediction paradigm rather than a controlling paradigm. This becomes evident by considering that the evolution of mobility patterns is not greatly influenced by the choice of the IoT device’s actions, but rather by the IoT device’s carrier actions. In fact, in the current state-of-the-art, the only influence an action makes is in finding contacts with different latency, which in fact drives the agent towards the best actions, given the reward and the mobility pattern, but does not directly “influence” the mobility pattern. This means that the only way to influence a mobility pattern, i.e. controlling it, could be to inform users about their patterns and make them change their behaviour: i.e. a feedback on a smartphone for a moving user or a new trajectory for a robotised IoT device.

The learning framework in fact better fits a policy evaluation environment, in which an agent follows a policy composed of a sequence of actions, but which is not directly influenced by the states reached. This implies that no policy improvement step is performed after every policy evaluation step, meaning that the sequence of actions is not modified according to the states reached over time. Evidently, in this thesis’s scenario, states are defined only by mobility patterns, therefore not greatly influenced by the actions taken, but only from the IoT device’s carrier. For example, in the case of a human-carried device, the social behaviour and the daily patterns of locations visited are the only factors that determine where and when the user will interact with other devices: i.e. its daily route to work.

Sutton’s temporal difference algorithm works by updating a value function at every step when following some policy $\pi$. The value function which results by following such a policy, named $V^\pi$, is updated backwards at every step. In particular, the value function update for state $s_t$ at step $t$ is:

$$V(s_t) \leftarrow V(s_t) + \alpha [R_t - V(s_t)],$$

(4.1)

where $\alpha$ represents the step-size parameter or learning rate (see Section 3.2.2) and $R_t$ is the reward obtained at step $t$. In its simplest form of 1-step update, the temporal difference reward at step $t$ becomes:

$$R_t^{(1)} = r_{t+1} + \gamma V(s_{t+1}),$$

(4.2)

where $r_{t+1}$ is the immediate reward after the transition between the current state $s_t$ and the next state $s_{t+1}$, $\gamma$ is the discount factor (see Section 3.2.2) and $V(s_{t+1})$ is the estimated value of the next state $s_{t+1}$. If $\delta_t$ is defined as the update the temporal difference learning has to perform at step $t$, in the case of a 1-step update, such an update becomes:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t),$$

(4.3)
leading to a value function update of:

\[ V(s_t) \leftarrow V(s_t) + \alpha \delta_t. \]  \hspace{1cm} (4.4)

Temporal difference methods are however not limited to a 1-step update but can be generalized to an \( n \)-step update. This leads to an algorithm, named TD(\( \lambda \)), which is given by Algorithm 2 below. The size of the update is controlled by a parameter \( \lambda \) for the so-called eligibility traces. This means that, in general, the update is not limited to the 1-step reward of Equation 4.2 but can be in general an \( n \)-step reward as such:

\[ R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}). \]  \hspace{1cm} (4.5)

This allows reaching an operation more similar to Monte Carlo algorithms where updates are based on the entire sequence of future rewards up until the ending/absorbing state. The eligibility traces are in fact a method to allow for averaged long-term rewards in multi-step updates to propagate back in time, based on a \( 0 \leq \lambda \leq 1 \) parameter. In such a case the \( \lambda \)-based reward becomes:

\[ R_t^{(\lambda)} = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}, \]  \hspace{1cm} (4.6)

in which the parameter \( \lambda \) works as an averaging decaying constant which gives “distant” updates smaller weights with respect to “closer” updates. In fact such a parameter influences the speed of the rewards decay, meaning that a lower value will have a fewer steps based update. In particular, if \( \lambda = 0 \), all the eligibility traces \( e(s) \) are 0 at step \( t \) except for the trace for
4.2 Learning Algorithms Based on Temporal Differences Methods

the current state $s_t$, which is equal to 1 (see line 9 and 12 of Algorithm 2). In such a case, denoted as TD(0), the update becomes the classical 1-step update of Equation 4.3 and the agent updates its value function estimates only by relying on the immediate reward and the next state estimate. Conversely, for any other value of $\lambda$, all the eligibility traces decay with $\lambda$ and more of the future rewards is used to update the current value function estimate. Evidently for $\lambda = 1$, the TD(1) algorithm becomes a way of implementing a Monte Carlo update, where such an update would be based on the entire trajectory in the states.

4.2.1 Function Approximation and Least Squares Temporal Difference Methods

When the state space is continuous or large, a more efficient way to learn and represent a value function is through function approximation. Since the interest is in predicting time representations, such as the arrival times and the departure times of IoT devices, such an approximation is adopted. This means that, instead of using a look-up table based function, the value function is completely defined by a set of parameters $\theta$ and a feature representation $\phi$:

$$V^\pi(s) \approx \theta \cdot \phi(s).$$  \hfill (4.7)

Under such a representation, the $\delta$ temporal difference update becomes:

$$\delta := \delta + \Delta \theta_t,$$  \hfill (4.8)

where the $\Delta \theta_t$ represents the temporal difference error and is computed as:

$$\Delta \theta_t = \left[ R_t + \gamma \theta^T \phi(s_{t+1}) - \theta^T \phi(s_t) \right] \sum_{k=1}^{t} \lambda^{t-k} \phi(s_k),$$  \hfill (4.9)

or, in a more compact manner:

$$\Delta \theta_t = e_t \left[ R_t + (\gamma \phi(s_{t+1}) - \phi(s_t))^T \theta \right],$$  \hfill (4.10)

where $e_t$ represents the eligibility traces and is equal to:

$$e_t = \sum_{k=1}^{t} \lambda^{t-k} \phi(s_k).$$  \hfill (4.11)

The $\delta$ temporal difference, updates the parameters for the function approximation as such:

$$\theta := \theta + \alpha \delta,$$  \hfill (4.12)
where \( n \) represents the *episode* considered, which is defined as one of the trajectories \((s_0, s_1, \ldots, s_T)\) in the state space until a terminal state \( s_T \) is reached. The parameters are in fact derived by performing a stochastic gradient descent on a cost function such as:

\[
J = \| \theta - \theta_\lambda \|^2. \tag{4.13}
\]

Minimizing such a cost function for deriving \( \theta_\lambda \) can be seen as solving a system of equations as such:

\[
d + C\theta_\lambda = 0, \tag{4.14}
\]

but without explicitly representing the \( d \) vector and the \( C \) matrix, which follow from the definition of the parameter update as:

\[
\theta := \theta + \alpha_n (d + C\theta + \epsilon), \tag{4.15}
\]

with \( \epsilon \) as a noise term. From Equation 4.10, it follows that:

\[
d = E \left[ \sum_{i=0}^{t} e_i R_i \right], \tag{4.16}
\]

and:

\[
C = E \left[ \sum_{i=0}^{t} e_i (\phi(s_{i+1}) - \phi(s_i))^T \right], \tag{4.17}
\]

where the expectations \( E \) are taken with respect to the distribution of trajectories.

While the classical temporal difference learning algorithm TD(\( \lambda \)) from Sutton is able to be used for prediction problems, it makes an inefficient use of the data and requires manual tuning of the step-size parameters, as discussed initially by Bradtke and Barto [134] and later by Boyan [135]. The Least Squares Temporal Difference algorithm LSTD(\( \lambda \)) instead overcomes such problems by constructing a vector \( b \) and a matrix \( A \) which, after \( n \) episodes, realizes an unbiased estimate of \( nd \) vector and \( -nC \) matrix. This allows retrieving the parameters when needed with a matrix inversion and a vector multiplication. While this could be seen as a complex operation, if the number of features is kept low as in this thesis’s case, the matrix and the vector have low dimensions. The Least Squares Temporal Difference algorithm is given by Algorithm 3. The objective of this algorithm is to learn the vector of the parameters \( \theta \) which approximates the value function. In order to perform such a task, the LSTD(\( \lambda \)) algorithm incrementally builds the least square estimates \( A \) and \( b \). Whenever needed, the parameters can be simply obtained through a matrix inversion (by Singular Value Decomposition) and a vector product as follows:

\[
\theta := A^{-1}b. \tag{4.18}
\]

Concerning the other parameters in Algorithm 3, \( e_t \) represents the eligibility traces, whose

1. Given: a simulation model for a policy \(\pi\); a featurizer \(\phi: S \rightarrow \mathbb{R}^K\) mapping states \(s \in S\) to feature vectors; a \(0 \leq \lambda \leq 1\) eligibility traces parameter;
2. Output: a parameter vector \(\theta\) for approximating \(V^\pi(s) \approx \theta \cdot \phi(s)\);
3. Set \(A := 0, b := 0, t := 0\);
4. for \(n := 1, 2, \ldots\) do
5. Initialize state \(s\);
6. Set \(e_t := \phi(s_t)\);
7. repeat
8. Take action \(a_t\), observe reward \(R_t\) and next state \(s_{t+1}\):
9. \(A := \mu A + e_t (\phi(s_t) - \gamma \phi(s_{t+1}))^T\);
10. \(b := \mu b + e_t R_t\);
11. \(e_{t+1} := \lambda e_t + \phi(s_{t+1})\);
12. \(t := t + 1\);
8. until state \(s_t\) is terminal;
9. end

update depth is influenced by the \(0 \leq \lambda \leq 1\) parameter. In addition, \(0 \leq \gamma \leq 1\) represents the discount factor which influences how much future rewards weight in comparison to more immediate rewards. Finally, \(0 \leq \mu \leq 1\) represents the exponential windowing factor (see Lagoudakis et al. [134]) which allows the algorithm to exponentially weight its incremental updates, therefore giving more weight to closer updates rather than past updates.

In conclusion, LSTD(\(\lambda\)) provides more efficient estimators in the statistical sense, which might require little more computation but, by building them incrementally, it does not require storing the trajectories, even when the state transitions are long. In addition, there is no requirement to adjust step-size parameters, which could affect convergence speed in other implementations. Finally, LSTD(\(\lambda\)) is not sensitive to the initial choice of the parameters or to the range of individual features, as is the case with the TD(\(\lambda\)) algorithm.

4.3 Arrivals and Departures Prediction Algorithm

The proposed arrival and departure times prediction (ADTP) algorithm covered in this thesis is based on two running instances of a LSTD(\(\lambda\)) algorithm. It is also assumed that every IoT device follows a certain mobility pattern given by the policy followed. Every action will therefore lead us to states represented as:

\[ s_{A_k} := k\text{-th arrival}, \ s_{A_k} \in S_A, \]

for the arrivals predictor and to:

\[ s_{D_k} := k\text{-th departure}, \ s_{D_k} \in S_D, \]
for the departures predictor. The value function can be therefore approximated as \( V^*(s) \approx \theta \cdot \phi(s) \), where the parameters and feature vectors can be set to:

\[
\theta_A = [\theta_{A_0}, \theta_{A_1}]; \quad \theta_D = [\theta_{D_0}, \theta_{D_1}]; \quad (4.21)
\]

\[
\phi_A = [1, \phi_A]; \quad \phi_D = [1, \phi_D]; \quad (4.22)
\]

where \( \phi_A \) and \( \phi_D \) represent the arrival times and the departure times at which the contact appears, as recorded by the IoT device.

It is the opinion of this thesis’s author that this representation for the value function, which values only arrival or departure times as features in order to predict future arrival or departure times, can also be expanded to tackle new features, as it is planned for future work. For example these could be metrics of popularity such as the number of interactions with a particular IoT device or metrics of social behaviour such as community membership or friendship as well as location tagging in order to build more complex knowledge about mobility patterns. However, this might require more complex non-linear function approximation in the parameters, which in turn could require use of advanced methods of representation.

Temporal Difference learning provides for a general multi-step prediction of a value representing a target for the learning process, which refines over the prediction over time. For example, it could be either the prediction of the weather over a finite number of days, which can be refined over time as new information becomes available, or the prediction of the time it takes for a small trip, which can also be refined over time as new information becomes available. Nevertheless, in the case of mobility patterns, the interest is only in predicting the next contact, therefore in performing a “one step ahead” prediction. To keep things simple and effective, a value function is learned for predicting the next arrival and departure times. Therefore, it is left for future work the case in which, between two consecutive contacts, the state evolutions and the feedback from the environment allow for a more accurate multi-step prediction, with refinement over time of the predictions for the next contact, as time elapses from the previous contact. In addition, since the value function learned will contain the explicit values of arrivals or departures, in case an evaluation of future multiple steps ahead is needed, this is possible by following the hypothetical trajectory in the state space.

In Figure 4.1 it is possible to see the prediction process for a policy evaluation framework. In every state the agent ends up into, a prediction about the next contact arrival and departure is made. When a one step ahead prediction is considered, the next predicted arrival or departure, intuitively does not depend (not even partially) on its next predicted arrival or departure. Since, in formulas, i.e. for arrivals:

\[
P_{S_{At}} = R_{t+1} + \gamma P_{S_{At+1}}, \quad (4.23)
\]
4.3 Arrivals and Departures Prediction Algorithm

![Diagram of state transitions for arrivals and departures](image)

**Figure 4.1: Prediction with Temporal Difference Learning.**

the discount factor is therefore set to $\gamma = 0$ to reflect the lack of such a dependency. Similarly, since a propagation of average rewards through eligibility traces is not needed to update previous state values with future rewards:

$$R_t = r_{t+1} + \lambda R_{t+1},$$  \hspace{1cm} (4.24)

the parameter is therefore set to $\lambda = 0$. In addition, the reward at step $t$ represents the actual value of the observed arrival or departure time. For example, for arrivals:

$$r_t = \phi_{A_t}.$$  \hspace{1cm} (4.25)

By considering the current state-of-the-art, one of the major issues is the capability to recognize when the mobility pattern changes its behaviour. For example, while in an office environment during weekdays the office is full of people carrying their IoT devices, the same office environment might be rather empty at night or during weekends. In addition, in order to have an algorithm which works with any mobility condition (i.e. controlled, public transportation systems based or human mobility based), the capability to adapt to any condition should be provided. The algorithm has therefore been equipped with the capability to recognize a sudden change in mobility patterns, intuitively recognizable by a lower accuracy on the predictions. In fact, a novel method to measure the accuracy of the predictor has been introduced in ADTP, which exploits a short error history of $N_E$ size (with $N_E = 10$ in this thesis’s case). At every
interaction with the environment, the error between the observed value and the previously predicted value is computed. At step $t$, such a prediction error becomes:

$$e_t = |\phi_{A_t} - P_{S_{A_t}}|.$$  (4.26)

In order to detect a change in the mobility pattern, a simple moving average of the error history is built in order to detect a sort of heteroscedastic trend in the error between the predicted and the observed actual values. In particular, every $\frac{N_E}{2}$ steps, for both the arrival and the departure predictor, the following moving average is computed:

$$E_{MA_t} = \frac{1}{N_E} \sum_{k=1}^{N_E} e_k.$$  (4.27)

The moving average is then compared with its previously computed value (at step $t - \frac{N_E}{2}$) and, if 50% higher in value, a dichotomy between the predictions and the actual observation is considered to exist. In such a case, a temporary “reset” for the exponential windowing factor $\mu$ introduced in Section 4.2.1 is provided. It is important to note that the value of 50% was selected after an evaluation with various values. In fact lower values have shown to trigger “resets” even when not necessary, while, vice-versa for higher values. Following the reset, the exponential windowing factor is therefore lowered to a $\mu_{min} = 0.3$ and subsequently incremented by $\Delta \mu = 0.1$ at every step until it reaches a maximum value of $\mu_{max} = 0.9$. This helps in the updates by weighting the previous $A$ matrix and $b$ vector estimates less, therefore incorporating newer information with a higher weight with respect to previous information. The values for the exponential windowing factor are selected based on a small evaluation that it was carried out, which showed a faster convergence to optimal predictions with such values.

### 4.4 Resource Scheduling based on Next Contact Predictions

ADTP’s resource scheduler leverages an arrival and a departure time predictor, in order to define a resource scheduling that is capable of optimizing both the power consumption and the latency of the discovery process for the next contact. In Figure 4.2 it is possible to see how the resource scheduler exploits the predictions and an error estimate about such predictions, in order to define the discovery schedule of a sensing device.

In order to achieve such an objective, predicted times are exploited to decide with which discovery schedule approach contacts when they are expected with either high or low probability. Similarly to CARD, the schedules are defined as slotted and customized asynchronous temporal

---

1Heteroscedasticity reports a condition when different statistical sub-populations with different variances are present.
overlap based discovery actions (see Chapter 2), since those protocols are deemed the most generally applicable in heterogeneous IoT scenarios. In particular, as in CARD, Disco is chosen as the baseline protocol for the scheduling of the actions, mainly for its practicality. Two types of schedules are defined:

- High Latency Schedule (HLS) which guarantees the discovery within a high latency bounded time $t_{\text{high}}$.
- Low Latency Schedule (LLS) which guarantees the discovery within a low latency bounded time $t_{\text{low}} \ll t_{\text{high}}$.

As in CARD, such latency bounds for discovery are defined based on the minimum contact duration which needs to be discovered. This means that, once the bound is set, the actions will discover with 100% probability all the contacts longer than such a bound. By naming the minimum contact duration as $D$ as in CARD (where this parameter is to be decided by application requirements), it is possible to define the latency bounds for the high latency and low latency schedule as:

- $t_{\text{high}}$ is set as 100% of the contact duration $D$ ($t_{\text{high}} = D$) for the high latency schedule,
- $t_{\text{low}}$ is set as 5% of the contact duration $D$ ($t_{\text{low}} = 0.05 \cdot D$) for the low latency schedule.

Note that, in the exact same way as for CARD, such a definition for the bounds allows to be sure that, with a low latency schedule, ideally 95% (in the ideal condition of no errors in the communication) of the contact time should be left after discovery. Similarly, with a high latency schedule, the bound allows to safely be aware of the contact discovery, thus always having a feedback from the environment. Evidently, such a definition also implies that contacts shorter than $t_{\text{high}}$ will not be guaranteed to be discovered with a 100% probability. This means that, in some situations (i.e. in human mobility patterns) a few contacts might be missed if the contacts are very short. This might cause problems in some applications, which however could
lower the minimum contact duration to be recognized and the latency bound autonomously, if needed, though eventually incurring a higher energy cost. However, since the predictions allow us to estimate both the next contact arrival and departure times (hence, also the duration as their difference), by simply letting the schedule to be adaptively decided (with some limits), such a parameter might be customized on-the-fly in future improvements.

Given the latency bounds $t_{\text{bound}} = t_{\text{low}}$ or $t_{\text{bound}} = t_{\text{high}}$ and the slot time $t_{\text{slot}}$, through Equation [3.3] the algorithm then computes a prime “candidate” value $p$ by considering the equivalence in the inequality, as in CARD. By building the Sieve of Atkin sequence of primes up until $p$ and picking the last two values (lower than $p$) as a balanced prime pair, a new and safer latency bound can be computed as:

$$t_{\text{bound}}' = p_i \cdot p_j \cdot t_{\text{slot}} \leq t_{\text{bound}}.$$  \hspace{1cm} (4.28)

Before describing the resource scheduling strategy, another parameter needed by such a scheduler is introduced in order to “track” the accuracy of the predictor at every step. In fact, at every step a feedback is received from the environment about how good the predictions are in comparison to the actual observed values. In ADTP, a prediction error is used to estimate the sparseness of such errors, thus having a numeric value representing the accuracy on a short history. By letting $\vec{\phi}_A$ representing the vector of the actual arrival times and $P_{S,A}$ representing the vector of the arrival times predictions, the estimated mean squared error is defined as:

$$MSE(P_{S,A}) = \mathbb{E}\left[(P_{S,A} - \vec{\phi}_A)^2\right] = \hat{\sigma}_e^2.$$  \hspace{1cm} (4.29)

Such a mean squared error is then computed on the previously discussed errors history as follows:

$$\hat{\sigma}_e^2 = \frac{1}{N_E} \sum_{k=1}^{N_E} \epsilon_k^2.$$  \hspace{1cm} (4.30)

The accuracy estimate, together with the predicted time of arrival and time of departure, contributes to defining the resource schedule an IoT device has to follow in order to provide an energy efficient and latency optimized discovery. In particular, the resource schedule is defined by a triple:

$$R_S = \langle t_A, t_D, \hat{\sigma}_e \rangle,$$  \hspace{1cm} (4.31)

where $t_A$ and $t_D$ are the next estimated arrival and departure times, output of the two predictors and $\hat{\sigma}_e$ is the square root of the mean squared prediction error over the error history. By relying on such parameters, a resource schedule for ADTP is designed as depicted in Figure [4.3]. In such a schedule, three phases are defined as follows:

- **First Phase**, to be scheduled when a contact with another IoT device is expected with a very low probability.
4.4 Resource Scheduling based on Next Contact Predictions

- **Second Phase**, to be scheduled when a contact with another IoT device is expected with a high probability.

- **Third Phase**, to be scheduled when a contact with another IoT device was not experienced in the previous first and second phases, hence following a miss due to inaccurate predictions.

As can be seen in Figure 4.3, the first phase is scheduled from the last departure time $t_{D_{k-1}}$ up until the next predicted arrival $t_{A_k}$ minus the square root of the mean squared prediction error $\hat{\sigma}_e$. The second phase is then scheduled right afterwards, up until the next predicted departure $t_{D_k}$. During either one of such phases, if a contact is discovered, a communication protocol is assumed established and data is exchanged between devices up until the contact ends. In such a case, when the contact ends, a new resource schedule is built by evaluating the arrival and departure predictors and new first and second phases are scheduled. Alternatively, if a contact is missed in both the first and second phases, a third phase is initiated by the device up until a new device is found, which then triggers a new first and second phase schedule. In order to optimize resources and provide maximum contact duration, HLS is scheduled, as defined before during both the first and the third phases. This helps to avoid energy wastage but still allows recognizing the eventual presence of nodes in the neighbourhood in case of errors in the prediction. In addition, an LLS is scheduled in the second phase, which allows a higher contact duration when contacts are expected with high probability.

In order to provide further power consumption reduction, a secondary feature called *selective sleeping* is introduced. This feature allows a complete sleep instead of a regular high latency schedule in the first phase. This allows a higher reduction in power consumption, but that could eventually lead to a reduction of the percentage of successful discoveries. To minimize such an effect and in order to make the number of misses negligible with respect to the number of contacts, a sleeping first phase is scheduled only if a contact is discovered during the previous second phase. This allows rewarding the discovery with less power consumption if the predictor’s accuracy was high during the previous contact. When the contact instead is discovered in the first or the third phases, a HLS based first phase is scheduled for the next contact, since the predictor’s accuracy has not been as high as expected. If the predictor has been very accurate, then, only LLS based second phases will be scheduled by ADTP.
A few corrective features are also introduced to avoid an unrealistic behaviour of the scheduler in certain prediction conditions. For example, when the accuracy is very high the $\hat{\sigma}_e$ term might tend to become closer zero. This might lead to a “drift” effect for which the predicted arrival times are found later and increasingly delayed. This means that the contacts might get shortened over time. For this reason, a minimum value for $\hat{\sigma}_e$ is introduced, as follows:

$$\hat{\sigma}_{e_{\min}} = p_{i_{LLS}} \cdot p_{j_{LLS}} \cdot t_{slot},$$

which is equal to the minimum time for a guaranteed discovery with low latency.

In addition, in a few situations in which contacts are quite short, the predictor might forecast a $t_D \leq t_A$, which would impossibly lead to a negative contact time and therefore to a zero duration second phase. To counteract such an effect, the arrival and departure times are averaged and half $\hat{\sigma}_{e_{\min}}$ is subtracted to derive the new arrival time and one $\hat{\sigma}_{e_{\min}}$ is added to the new arrival time to derive the new departure time. Therefore, if $t_D \leq t_A$ the new arrival time becomes:

$$t_{A_{\text{new}}} = \frac{t_A + t_D}{2} - \frac{\hat{\sigma}_{e_{\min}}}{2},$$

and the new departure time becomes as follows:

$$t_{D_{\text{new}}} = t_{A_{\text{new}}} + \hat{\sigma}_{e_{\min}},$$

therefore mitigating the error in the prediction, which forecasts a departure before an arrival.

### 4.5 Conclusions

In this section, ADTP, an Arrival and Departure Time Prediction and Discovery framework which introduces a new learning and prediction algorithm for arrival and departure times in IoT scenarios for opportunistic networking is illustrated. Different from the current state-of-the-art solutions, ADTP introduces the possibility to predict numeric values about the arrival and departure times, therefore introducing numeric estimates about the time needed to be waited for next contact arrivals and about the durations of such future interactions.

The prediction algorithm allows efficient planning of the discovery and the communication process for the next expected contact. In fact, a resource allocation scheme based on asynchronous discovery protocols is introduced in ADTP in order to optimize the discovery process to obtain lower latency and energy expenditure. Indeed, ADTP can reduce power consumption with respect to the current state-of-the-art and it provides a latency optimized discovery which allows for the possibility to exploit most of the contact duration.

One of the novelty of ADTP is the possibility to track the accuracy of the predictions with respect to the actual observed values. This helps to recognize eventual abrupt changes in mobility patterns which would cause the errors to increase substantially over a certain finite
window of observations. In addition, the accuracy estimates also help to define the resource schedule, which can therefore be tailored to the uncertainty of the predictor, to reduce the number of misses.

Furthermore, ADTP does not require any adjustment of its parameters according to changing mobility conditions and does not require measuring in advance or providing additional parameters to derive the resource allocation. In addition, ADTP is largely applicable, due to its use of asynchronous, temporal overlap based, latency bounded discovery protocols (i.e. Disco) combined in a learning framework which requires few computational capabilities (i.e. the LSTD($\lambda$) algorithm). This is indeed a desirable property in IoT scenarios of opportunistic networking in which heterogeneous IoT devices need to discover and interact with each other. The LSTD($\lambda$) algorithm, in fact requires only a two-by-two matrix inversion and a vector multiplication which makes it computationally efficient and applicable to many IoT devices.

In addition, different from the current state-of-the-art, memory requirements for such an algorithm are very low, since just the least squares estimates and the function approximation parameters need to be stored in memory. Moreover, such an algorithm requires no training and it converges quite rapidly with few interactions with the environment, as well as being a more efficient estimator in a statistical sense, which builds estimates incrementally without storing all the trajectories. In addition, it does not require adjustment of the step-size parameters or an accurate initial choice of the parameters as in previous learning algorithms, and it is less sensitive to the range of individual features.

Finally, the prediction framework allows not only to derive estimated time of arrivals and durations for the next contacts, but also allows predicting multiple steps ahead. This allows an application to plan its discovery and communication not only for the next contact, but also for future contacts. Potentially, and as it will be discussed also in Chapter 7, this means that, as a future extension of ADTP, short unmeaningful contacts could be discarded in lieu of more favourable future contacts, and communication sessions can be planned and scheduled according to future predicted contacts.
Chapter 5

Implementation

This chapter introduces the implementation strategy which is adopted for the evaluation of the proposed contributions. After an overview of the Network Simulator 3 (NS-3) which has been used for the simulations, a review of the necessary extensions to this network simulator is presented. In particular, an application which reproduces a relevant state-of-the-art framework for Resource Aware Data Accumulation is firstly presented. Then, this thesis’s first contribution for Context Aware Resource Discovery (CARD) is provided in detail. An introduction to a Python-based framework for Reinforcement Learning is then reported, along with the proposed extensions necessary to simulate this thesis’s second contribution, i.e. Arrival and Departure Time Prediction (ADTP). Finally, an overview of the implementation of the Arrival and Departure Time Prediction and Discovery framework under the NS-3 environment is reported.

5.1 Introduction

The aim of this implementation is to evaluate the contributions of this thesis against the state-of-the-art solutions in order to benchmark their performance under realistic IoT scenarios of opportunistic networking. In order to achieve such an objective, a network simulator has been used. NS-3 [137] has been in fact selected for various reasons:

- it is an actively developed simulator with many readily available modules which can be used and extended to suit the simulating needs,

- it provides pre-built and extensible mobility models that are needed in order to simulate nodal movements, thus allowing to create complex IoT scenarios for opportunistic networking,

- it features an energy model which has been extended to analyse power consumption during the evaluation of the implemented discovery protocols,
• it provides for logging tools and implements its modules completely in C++, thus allowing for evaluation of complex machine learning algorithms.

In fact, by being completely open source, customizable and extensible, NS-3 allows evaluating learning algorithms that require external linear algebra libraries, which are linked into the framework, as explained in the next sections.

Furthermore, the Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network (PyBrain [138]) library has been used in order to simulate advanced reinforcement learning algorithms. In particular, the PyBrain’s environment has the following benefits:

• it provides many recent reinforcement learning algorithms and classical scenarios,
• it allows the use of advanced reinforcement learning features, such as experience replay and function approximation,
• it has the possibility to integrate data for evaluation purposes quickly into the framework, thanks to the wide availability of Python’s libraries.

In fact, the use of such a library has allowed avoiding long simulation times and quickly evaluating learning algorithms by just focusing on data, rather than on the simulator’s implementation of every network module.

5.1.1 Network Simulator Overview and Extensions

The Network Simulator 3 (NS-3 [137]) is a discrete event simulator written in C++. The simulator is organized as a library which can be linked by complex simulation scripts in which the network topology and the simulation parameters can be defined. Due to Python’s bindings of the C++ simulator APIs, such simulation scripts can either be written in C++ or in Python, thus allowing to be easily included in complex scripts. The simulator framework provides for many basic and advanced libraries for implementing different networking models and functionalities. In Figure 5.1 it is possible to see the main modules provided by such NS-3 libraries. The main Core module provides for the NS-3 simulator basic functionalities, which are:

• Attributes for accessing and organizing parameters ranges and values of the models.
• Callbacks for wrapping functions or objects.
• Command Line Parsing and System Services to interact with OS calls and to input simulation parameters.
• Debugging and Logging as well as Error Handling tools.
• Object Base classes and Smart Pointers for memory management and object aggregation.
• Scheduler and Events management as well as Simulator and Time arithmetic control.
5.1 Introduction

• **Random Variables** for various random distribution generators.

• **Tracing** and **Testing** classes for collecting traces and testing functionalities.

The **Network** module, instead provides for the basic networking functionalities, which are:

• **Address** abstraction (i.e MAC, IPv4 and IPv6).

• **Channel** and **Data Rate** as well as **Error Model** abstractions.

• **Nodes** and **Network Device** abstractions.

• **Packet**, **Queue** and **Socket** abstractions.

Moreover, the **Internet** module provides for basic Internet Protocols implementations, such as:

• **Address Resolution Protocol** (ARP).

• **Internet Protocol version 4** (IPv4).

• **Internet Protocol version 6** (IPv6).

• **Transmission Control Protocol** (TCP).

• **User Datagram Protocol** (UDP).

The **Mobility** module instead introduces several mobility models, such as **Random Walk** and **Random Waypoint**, or the possibility to follow synthetic customized traces written according to the NS2 traces language [139]. In addition, **Applications** for traffic generation and data sinks could be associated to nodes. Routing modules are also provided, such as:

• **Ad hoc On-Demand Distance Vector (AODV)** [140].
- Click Modular Router Integration [141].
- Destination-Sequenced Distance Vector (DSDV) [142].
- Dynamic Source Routing (DSR) [143].
- Neighbour-Index Vector (NIX-Vector) routing [144].
- Optimized Link State Routing (OLSR) [145].

Different NetDevices implementation are also provided, such as, i.e. CSMA, Bridge, Point-To-Point, Mesh, OpenFlow Switch, LTE, Wi-Fi and Wi-Max. Additional modules for Statistics such as Data Aggregators and plotting with GnuPlot [146] are provided, together with many Utils such as Network Animation, Flow Monitor, MPI Distributed Simulation and Helper classes to aid in building complex simulation scripts and topologies. Finally, Energy Models and Propagation Models are provided in order to simulate realistic behaviours.

In Figure 5.2 it is possible to see the networking model of NS-3, which allows communication between two distinct nodes. Every Node abstraction has associated with it one Application or more. Applications on different nodes can communicate with each other through a Socket which the Application handles, as it would happen in any real world application. A Packet generated by such applications transverses the networking stack, is encapsulated with relevant protocols (i.e. TCP and IPv4) and is eventually routed until it reaches the destination node. The message is transmitted via the relevant NetDevice (i.e. a Wi-Fi device) and sent on a Channel to the destination node, which will receive it and forward it to the relevant application.

Since the main objective of this implementation is to evaluate this thesis’s contributions in an IoT scenario of opportunistic networking where IoT devices are heterogeneous and may be
equipped with several radios, a custom implementation of the Channel and NetDevice classes has been introduced. In addition, a customization of the Energy Model with the objective to provide for a way to efficiently measure power consumption has been performed.

In Table 5.1 it is possible to see the parameters with which the LossyChannel has been implemented, by inheriting from the NS-3 Channel abstract base class. Such an implementation,

Table 5.1: NS-3 Attributes for customized LossyChannel.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default Value</th>
<th>Member Variable</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropagationLossModel</td>
<td>Pointer</td>
<td>N/A</td>
<td>m_loss</td>
<td>N/A</td>
</tr>
<tr>
<td>PropagationDelayModel</td>
<td>Pointer</td>
<td>N/A</td>
<td>m_delay</td>
<td>N/A</td>
</tr>
<tr>
<td>PropagationFadingModel</td>
<td>Pointer</td>
<td>N/A</td>
<td>m_fading</td>
<td>N/A</td>
</tr>
<tr>
<td>EnergyDetectionThreshold</td>
<td>Double</td>
<td>-90</td>
<td>m_edThreshold</td>
<td>dBm</td>
</tr>
<tr>
<td>TxGain</td>
<td>Double</td>
<td>0</td>
<td>m_txGain</td>
<td>dB</td>
</tr>
<tr>
<td>RxGain</td>
<td>Double</td>
<td>0</td>
<td>m_rxGain</td>
<td>dB</td>
</tr>
<tr>
<td>RadioRange</td>
<td>Double</td>
<td>100</td>
<td>m_rangeMax</td>
<td>m</td>
</tr>
<tr>
<td>TxPowerLevels</td>
<td>Uinteger</td>
<td>26</td>
<td>m_nTxPower</td>
<td>N/A</td>
</tr>
<tr>
<td>SelectedPowerLevel</td>
<td>Uinteger</td>
<td>26</td>
<td>m_powerLevel</td>
<td>N/A</td>
</tr>
<tr>
<td>TxPowerStart</td>
<td>Double</td>
<td>-25</td>
<td>m_txLevelStart</td>
<td>dBm</td>
</tr>
<tr>
<td>TxPowerEnd</td>
<td>Double</td>
<td>0</td>
<td>m_txLevelEnd</td>
<td>dBm</td>
</tr>
</tbody>
</table>

in fact, works as a wireless channel in which it is possible to attach one of the propagation loss, fading and delay models of the NS-3 Propagation module. In addition, it is possible to customize the transmission and reception gains (dB) of the antennas, the energy detection threshold of the receiver and the radio range (m) after which a complete cut-off of the communication is in place. The radio output power (dBm) is defined as a particular transmission level (i.e. SelectedPowerLevel) out of all the possible transmission levels (i.e. TxPowerLevels) in which the admissible range of output power is divided into (from TxPowerStart to TxPowerEnd). A LossyNetDevice implementation has also been provided as an interface to the higher levels of the stack, in which the only attribute implemented is a packet loss model ReceiveErrorModel, modelled as a pointer to an NS-3 Error model stored in the m_receiveErrorModel member variable. A LossyContainer and a LossyHelper class have also been implemented in order to have a more agile instantiation of the channel and the netdevices in the simulation scripts. The helper creates a LossyChannel to which it attaches a LogDistance Propagation loss model and a NakagamiFading model, as well as setting the other parameters based on the IoT device radios and antennas considered (i.e. from CC2420 or CC1000 datasheets \[147, 148\]). It then creates a LossyNetDevice for every Node considered (grouped inside a NodeContainer) and aggregate the objects to the relevant nodes.

The NS-3 Energy Model \[149\] refers to the situation of Figure 5.3 where a DeviceEnergyModel which models a component’s power consumption is updated through the ChangeState member function. In order to model also complex devices with multiple devices, the NS-3 energy model provides for a separate class which models the energy source, such as, i.e. a battery. In order to provide a customized implementation, a child class for the energy model for
a generic radio (RadioEnergyModel) has been implemented. In addition, a basic energy source (BasicEnergySource) has been exploited, since modelling more complex behaviour (i.e. Li Ion Batteries) is not in this thesis's objectives. The RadioEnergyModel offers three attributes which model three possible current consumption states:

- **StandbyCurrentA**, modelled as a double member variable named \( m_{\text{standbyCurrent}} \).
- **RxCurrentA**, modelled as a double member variable named \( m_{\text{rxCurrent}} \).
- **TxCurrentA**, modelled as a double member variable named \( m_{\text{txCurrent}} \).

Such attributes are modelled based on the IoT device’s radio power consumption, thus relying on relevant datasheets.

In order to evaluate the contributions of this thesis in different mobility scenarios, some functions have been created in the main simulation scripts, which are capable either of creating synthetic traces or parse real world traces in order to create NS-2 language compliant traces. The synthetic traces generated, include:

- **Deterministic** traces which consists of moving at a fixed speed a mobile node which interacts periodically with a statically deployed node.

- **Multiple Deterministic** traces which consists of the same Deterministic scenario of above, though in which the inter-contact times are increased or decreased in steps.

- **Gaussian** traces which consists of the Deterministic traces in which the inter-contact time is drawn at every iteration from a Gaussian Distribution with fixed mean and variance values.

![Figure 5.3: NS-3 Energy Model.](image-url)
• **Multiple Gaussian** traces which consists of Gaussian traces as above, though in which the distribution mean representing the inter-contact time is increased in steps as in the Multiple Deterministic trace.

It is possible to see in Figure 5.4 the temporal evolution of contacts according to the synthetic traces. The Deterministic scenario sees a periodic contact, the Multiple Deterministic sees a variation over time in steps and the Gaussian and Multiple Gaussian, instead see a contact normally distributed within a certain variance, represented by the bell-shaped distribution. A parser to extract information from traces collected during a local experiment [150] has also been developed. These traces include Bluetooth mobility patterns of interaction between smartphone’s carriers and deployed infrastructure in an office environment, as well as Passive Infrared Sensors based presence detection. Finally, the simulations have also been evaluated against the real world mobility traces datasets of the Haggle project [151], which are used as a benchmarking reference by many authors in literature. These traces include Bluetooth sightings by users carrying small IoT devices (iMotes) for six days in the Intel Research Cambridge Lab and Computer Lab at University of Cambridge as well as during the IEEE INFOCOM 2005 conference. The synthetic and real world traces have then been used with the NS2MobilityHelper provided by the NS-3 simulator, which parses the traces and makes the corresponding nodes move accordingly.

Finally, a synthetic mobility model such as STEPS [152], which models advance features such as:

• a preferential location attachment, which models the probability of the distance travelled as inversely proportional to such a distance,

• location attractors, which model the probability to move closer to certain locations,

has been implemented. This model generates traces following a truncated power law distribution for the survival function of the inter-contact times, which real traces have shown to follow in previous research [10]. In particular, two power-law distributed random variables named AttractorDistanceRandomVariable and StayingTimeRandomVariable which inherit from RandomVariableStream have been implemented. Both random variables take three attributes:
• Min a double value representing the lower bound on the values returned by this stream.

• Max a double value representing the higher bound on the values returned by this stream.

• Alpha or Tau, which are double values representing the exponents for these power law distributions (see 162 for more details).

The StepsMobilityModel class has then been implemented, with the attributes reported in Table 5.2. In STEPS, the networking area is divided into a \( N \times N \) square torus in which the nodes can move, where \( N \) is the GridSize attribute. Every node has an initial squared zone \( Z_0 \) of coordinates \((AttachmentX, AttachmentY)\) within which it is deployed, with dimensions equal to ZoneWidth. At every iteration, the mobility model draws a distance from the power-law distributed AttractorDistanceRandomVariable with \( \alpha \) exponent AttractorPower. The algorithm then selects randomly between all the zones at the distance just found, according to the Distance random variable, thus finding the destination zone \( Z_i \) at iteration step \( i \). By using the \((SpatialX, SpatialY)\) random variables, the algorithm then selects random coordinates within the \( Z_i \) zone and performs a linear walk, with a speed drawn from the Speed random variable, from the previous coordinates to these new coordinates (i.e. from within \( Z_0 \) to within \( Z_1 \)). The algorithm then selects a power law distributed time from the StayingTimeRandomVariable with exponent equal to TemporalPreference and distributed within TimeLimitMin and TimeLimitMax. Then, it performs Random Waypoint movements for the time just drawn within the \( Z_i \) zone, selecting from Speed and Pause random variables. Finally, the algorithm iterates by drawing at every step a new destination zone, and runs for a time equal to RunningTime, after which it stops moving.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default Value</th>
<th>Member Variable</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>GridSize</td>
<td>Uinteger</td>
<td>20</td>
<td>m_gridSize</td>
<td>N/A</td>
</tr>
<tr>
<td>AttachmentX</td>
<td>String</td>
<td>UniformRandomVariable</td>
<td>m_axxRV</td>
<td>N/A</td>
</tr>
<tr>
<td>AttachmentY</td>
<td>String</td>
<td>UniformRandomVariable</td>
<td>m_ayyRV</td>
<td>N/A</td>
</tr>
<tr>
<td>ZoneWidth</td>
<td>Double</td>
<td>120</td>
<td>m_zoneWidth</td>
<td>m</td>
</tr>
<tr>
<td>AttractorPower</td>
<td>Double</td>
<td>0</td>
<td>m_alpha</td>
<td>N/A</td>
</tr>
<tr>
<td>Distance</td>
<td>String</td>
<td>UniformRandomVariable</td>
<td>m_distRV</td>
<td>m</td>
</tr>
<tr>
<td>SpatialX</td>
<td>String</td>
<td>UniformRandomVariable</td>
<td>m_sxxRV</td>
<td>m</td>
</tr>
<tr>
<td>SpatialY</td>
<td>String</td>
<td>UniformRandomVariable</td>
<td>m_syyRV</td>
<td>m</td>
</tr>
<tr>
<td>Speed</td>
<td>String</td>
<td>UniformRandomVariable[Min=3.6][Max=40.0]</td>
<td>m_speed</td>
<td>km/h</td>
</tr>
<tr>
<td>TemporalPreference</td>
<td>Double</td>
<td>0</td>
<td>m_tau</td>
<td>N/A</td>
</tr>
<tr>
<td>TimeLimitMin</td>
<td>Double</td>
<td>20</td>
<td>m_minTimeLimit</td>
<td>s</td>
</tr>
<tr>
<td>TimeLimitMax</td>
<td>Double</td>
<td>30</td>
<td>m_maxTimeLimit</td>
<td>s</td>
</tr>
<tr>
<td>Pause</td>
<td>String</td>
<td>UniformRandomVariable[Min=1][Max=5]</td>
<td>m_pause</td>
<td>s</td>
</tr>
<tr>
<td>RunningTime</td>
<td>Double</td>
<td>864000</td>
<td>m_stopTime</td>
<td>s</td>
</tr>
</tbody>
</table>

Finally, a StepsMobilityHelper has been implemented in order to configure the mobility model according to the topology.
5.2 Resource Aware Data Accumulation

In order to compare performance against the state-of-the-art, RADA has been implemented in NS-3 as a child class of the NS-3 Application class and then has been installed on nodes. In Figure 5.5 it is possible to see the main steps that the RADAApplication performs during its execution.

Firstly, the simulator schedules the RADA application to start (A) on a node to which it is aggregated. After the relevant object is constructed, the application also initializes (B) the RadioEnergyModel and the BasicEnergySource for such a node. A PacketSocket is then created (C), bound, set as broadcast and connected, setting relevant member functions as callbacks for Connect, Accept, Receive and Close events. In particular, the HandleRead method (D) has been set to process packets received through the socket from other applications through the LossyNetDevice and the LossyChannel. Such a method, records the simulation times at which a discovery packet (beacon) is received, as well as the inter-contact times and the latencies with which it is received with respect to the time of initial contact.

The learning process is then initialized (E) by creating three RADATask objects, added to a tasklist, representing the three duty cycling actions which RADA schedules over time. A new initial state is created as a RADAState object, which is initialized in its Q-values for all the actions and added to the stateslist. The current state pointer is then assigned to the initial state just created, whereas the current action pointer is set to null. The application then schedules a MobilityChecker (F) which periodically checks the distance from the current node to the other nodes. If the start or the end of a contact is detected, appropriate flag variables are set and logging variables are initialized or updated. These include, but are not limited to latency, energy outside and inside contacts, number of discoveries, residual contact time and contact duration. At the same time, a TimeDomain is scheduled (G), which controls the steps of the Q-Learning between actions executions.

In the TimeDomain update, at the first iteration (H), the learning update step is skipped since no actions have been executed before. A new exploration factor is then retrieved (I) according to the Equation 3.6 which allows to randomly draw between exploration and exploitation actions. According to the result of the draw at this step (J), either a random action (K) is selected as the current action coherently with the exploration strategy, or the best action (L) is selected as the current action according to the Q-values for the current state. Such a current action is then scheduled for execution (M), which, in this thesis’s case involves the execution of Disco actions instead of RADA’s duty cycling actions in order to have a fairer comparison with RADA, thus evaluating only the performance of the learning framework.

The execution of Disco actions involves the strategy mentioned in Section 3.2.1. In particular, two counters for the prime numbers have been used and reset according to the action selected. After every update of the prime counters, either a waiting slot or an awake slot is scheduled for the slot time duration. The awake slot schedules two beacon transmission, one
at the beginning and one at the end of the slot, as well as a listening phase in between the two beacon transmissions. This means that, during the awake slot, the radio states of the energy
model are changed accordingly to such a schedule and that two packets are scheduled and sent through the socket. In the waiting slot, instead, a standby radio state is scheduled in which the radio is powered off.

At the beginning of the action execution, a new TimeDomain update is then scheduled after relevant time. After the execution of the action, at the new TimeDomain evaluation, a new state \((N)\) is created according to the evaluated state variables. In addition, the Hamming distance of the newly created state is evaluated towards all the states in the states list: if such a distance is higher than the Hamming threshold, the new state is discarded and the similar state is used instead; otherwise, the new state is added into the states list. The reward \((O)\) for the executed action is then computed with the relevant equation and a new update for the Q-Learning is then computed \((P)\). Furthermore, the Q-value of the previous state is updated \((Q)\) and a new action is scheduled according to the exploration/exploitation trade-off. Finally, at relevant time, the simulator schedules the RADA application to stop \((R)\) and clean everything, as well as logging the global variables which require computation such as total energy, discovery ratio and cumulative residual contact time.

### 5.3 Context Aware Resource Discovery

In order to evaluate this thesis’s first contribution, CARD has been implemented as a derived class from the Application parent class. In addition, a \(CARDAction\) and a \(CARDState\) class have been implemented as derived classes of the NS-3 Object base class, with the objective of representing the Q-Learning actions and states. It is possible to see in Figure 5.6 the main steps of the application execution, which conceptually is very similar to RADA, since they share the same Q-Learning algorithm.

At the beginning \((A)\) of the execution, the CARD Application initializes \((B)\) the RadioEnergyModel and the BasicEnergySource for the node in which such an application is installed. Similarly to RADA, a PacketSocket is created \((C)\), bound, set as broadcast and connected, setting relevant member functions as callbacks for Connect, Accept, Receive and Close events. A HandleRead method \((D)\) similar to the one of RADA has also been implemented in order to process packets received through the socket from other applications through the LossyNetDevice and the LossyChannel. Such a method records the inter-contact times and the latencies with which a beacon is received.

The application then initializes \((E)\) the learning process and the actions parameters, such as the number of sub-actions (see Equation [3.11]) and the sub-actions durations. By exploiting such parameters in combination with the periodicity and the minimum contact duration, a CARDAction object for every possible schedule containing those informations, is thus created and added to an actions list. The initial CARDState object which represents no contacts found is then created and initialized, as well as added to the states list. Similarly to RADA, a MobilityChecker \((F)\) is also scheduled in parallel, which checks the distances from nodes and
sets logging variables accordingly. An exploration factor is then drawn (G) according to the fixed exploration policy and either a random action (H) or the best action (I) for the current
state is selected as the current action to be scheduled. The application then plans the action’s execution (J), according to the action’s parameters and the sub-actions schedules. Firstly, a certain number of high latency sub-actions (K) are scheduled according to the current action definition. Then, if the action is not based on only high latency sub-actions, a low latency sub-action (L) is also executed. Eventually, remaining high latency sub-actions (M) are scheduled until needed. Moreover, if a discovery is made during one of the relevant sub-actions, the application schedules a sleeping action (N) up until the end of the action scheduled.

At the end of the action execution, the resulting state is evaluated by looking at the beacon reception pattern during the action execution (O). If the state is already present in the states list, the corresponding state is selected as the new state. Alternatively, a new state is created and added to the states list. The reward is then computed (P), together with the Q-Learning update (Q) and the Q-value for the previous state is thus updated (R). Finally, a new iteration is started and a new action is executed according to the exploration strategy up until the simulation has ended. When the simulation time is stopped (S), the final metrics are then computed and logged.

5.4 Arrival and Departure Time Predictor

In order to concentrate on the evaluation of the prediction algorithm for arrival and departure times with real world and synthetic data, the Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network (PyBrain) library [138] has been used.

PyBrain is a modular and easy to use library providing with recent state-of-the-art machine learning algorithms for research purposes. Artificial Neural Networks can be easily built by adding different units and connecting them with each other in feedforward or recurrent architectures. Many supervised learning techniques for training such networks are provided, such as Back-Propagation, R-Prop, Support-Vector-Machines (LIBSVM interface), Evolino, Momentum and Natural Gradients. In addition, data that needs to be fed to such architectures can be preprocessed with unsupervised techniques, such as K-Means Clustering, Principal Component Analysis, Locality Sensitive Hashing and Deep Belief Networks. Moreover, black-box optimization techniques are provided to be used in problems which reduces to the minimization of a cost function, such as (Stochastic) Hill-climbing, Particle Swarm Optimization, Evolution Strategies, Genetic Algorithms, Covariance Matrix Adaption and Multi-Objective Optimization.

Reinforcement Learning temporal difference methods such as Q-Learning, SARSA, Neural Fitted Q-iteration are also implemented. Policy Gradient based techniques for continuous spaces are provided, such as REINFORCE and Natural Actor-Critic algorithms. Finally, different exploration strategies are provided, starting from $\epsilon$-greedy to Boltzmann, Gaussian or State-Dependent strategies. Many classical environments in which it is possible to try the learning outcomes are also provided, such as Mazes, 3D Environments, Games, Pole-Balancing and Car-Racing. PyBrain’s main modules and interface diagram for the Reinforcement Learning
framework are reported in Figure 5.7. The Environment module models a real world environment for a Markov Decision Process (MDP), in which the agent tries different actions in order to transition between states and obtain rewards for its actions. The Agent module, represents the reinforcement learning agent and is composed by four blocks, named Module, Explorer, Learner and Dataset. The Agent is interfaced with the Environment through a Task module, which has the objective of solving scaling and normalization issues of Observations and Actions between the agent and the MDP. It also specifies a goal for the environment and how the Reward is given to the Agent. After every action execution, within the agent an Observation arrives to the Module, which has the objective of transforming it into an Action, which is fed to the Explorer module. The Explorer is an optional module which controls the behaviour of the Action according to the strategy selected. The triples composed by Observation, Action and Reward are then stored in the Dataset module, thus allowing for complex iterations such as experience replay. Finally, the Learner module collects data from the datasets, either after a certain number of steps or at the end of an episode and modifies the parameters of the Module accordingly.

By relying on such a framework, some extensions to the current reinforcement learning modules have been developed in order to experiment with the arrival and prediction framework. A new PolicyEvaluationEnvironment class derived from the Environment class has been implemented in order to model a reinforcement learning MDP in which the observations sequence is known, since it comes from mobility patterns data. In particular, getSensors() and performAction() methods have been adapted to such an environment. An ArrivalTask class derived
from the Task class has also been implemented in order to interface data coming from the PolicyEvaluationEnvironment. In particular, the getReward() and getObservation() methods have been implemented in order to translate the arrival and departure times observations from the environment into feature vectors and to pass the reward to the agent.

![Figure 5.8: Arrival and Departure Time Prediction Main Program.](image)

In order to create a vector with the sequence of observations (i.e. arrival or departure times) a MobilityParser class has been implemented for parsing the real world mobility traces collected...
during the in-house experiment \cite{150} and from the Haggle traces. The getArrivalTimes, getDepartureTimes methods have been implemented, as well as the getIntercontactTimes and the getDurations methods for retrieving different information from the traces. Similarly, a SyntheticTrace class has been implemented with deterministicTimes, gaussianTimes, multipleDeterministicTimes and multipleGaussianTimes methods for generating synthetic mobility traces based on relevant parameters such as: inter-contact time, contact duration, number of days, increment and standard deviation. PyBrain’s LinearFA Agent module has also been modified in order to interface it with the ArrivalTask, in particular in the integrateObservation method.

The LSTDQLambda class is used as the main learner module and modified according to the needs of the LSTD algorithm. In particular, relevant parameters are adjusted, the actions space is reduced to only one possible action to emulate a policy evaluation behaviour and an exponential windowing factor is introduced, together with the errors history, the moving average computation and the logging functionality.

In Figure 5.8 it is possible to see the evolution of the main steps of the arrival and departure time predictor. First, after the application starts (A), either the real world traces are parsed or the synthetic traces are generated (B), with the corresponding observation vectors of data. Then, a PolicyEvaluationEnvironment which takes as input such a vector is created (C). An ArrivalTask is then generated (D) from the current environment, which takes as input also the grade of the polynomial for the feature vectors to be crafted (in this thesis a first order polynomial has been used). The LSTDQLambdaLearner (E) is then created and its parameters initialized. In addition, a Linear FA Agent is connected (F) to such a learner. Moreover, an Experiment (G) tying together the agent and the task is then created. The algorithm is then iterated (H) along the data and the parameters updated according to the learner. Finally, by leveraging Python’s libraries capabilities, results are stored in NumPy (a fundamental package for scientific computing) styled arrays and plotted (J) with matplotlib (a plotting library capable of providing a MATLAB-like interface).

5.5 Arrival and Departure Time Scheduler Planner

In order to evaluate in a realistic environment and to evaluate the performance of a discovery framework based on the ADTP algorithm, an implementation has been made under the NS-3 simulation environment. The Armadillo C++ library \cite{153} has been linked to the NS-3 framework in order to implement the necessary matrix/vector computation needed by the LSTD learning algorithm. Such a library has been chosen in order to avoid replication of matrix-vector multiplication or matrix inversions operations, which could however be easily coded in any IoT device, without requiring the use of any advanced linear algebra C++ library.

In Figure 5.9 it is possible to see the main ADTP application steps. As in the previous application implementations for RADA and CARD, the first part of the application is dedicated to the initialization of the RadioEnergyModel and the BasicEnergySource (A-B). In addition, the
PacketSocket (C) is created, bound, set as broadcast and connected, setting relevant member functions as callbacks for Connect, Accept, Receive and Close events. The HandleRead method (D), as in previous applications, processes packets received through the socket, records latency and inter-contact times, and updates the arrival and departure schedules when a discovery is
made. Two ADTPPredictor objects are then created (E) for predicting arrivals and departures according to the LSTD algorithm and a MobilityChecker is instantiated (F). Such predictors are then initialized in the parameters vector, least square estimates, error history and moving average, as well as state vectors and eligibility traces.

In order to start the discovery, a resource schedule is retrieved by asking the predictor for the next arrival and the next departure as well as the mean square error through relevant get methods (G1-G2). A resource schedule is thus created and used to schedule, in order, a first high latency (or sleep) phase (H) and a second low latency phase (I) for relevant durations as given by the resource schedule. If during such phases a discovery is made (J) through the HandleRead method, the arrival predictor is updated (K) through an iteration of the algorithm. After the contact is ended, the departure schedule is also updated (L) and a new resource schedule is thus obtained as before (G2). The predictor update involves updating the state vector with the new state, the eligibility traces, the errors history and the mean squared prediction error, as well as computing the moving average and triggering a “reset” in the exponential windowing factor if an abrupt change in the mobility patterns is detected. The least square estimates are then computed and the parameters vector is updated accordingly. Finally, if a contact miss is recorded or if a contact has not been found during the first two phases, a third high latency phase (M) is scheduled until a contact is found. As in previous applications, when simulation time stops (N), the final metrics are then computed and logged.

5.6 Conclusions

The implementation of the contributions allowed the evaluation under different mobility scenarios, as well as to build the necessary extensions for computing performance metrics and quickly analyse the results with graphs. Thanks to NS-3’s mobility and energy models, previous relevant state-of-the-art and this thesis’s contributions have been developed with the objective of their evaluation. Moreover, by using Python’s PyBrain library, toy scenarios for reinforcement learning in which to simulate with either synthetic or real world mobility data have been developed, in order to quickly evaluate novel algorithms. Finally, due to the general C++ based NS-3’s open architecture, this thesis’s contributions have been easily integrated within the framework by relying on widely available linear algebra C++ libraries.
Chapter 6

Evaluation

This chapter reports the results of the performance evaluation performed under the simulation frameworks. After an introduction about the aim of the evaluations, a validation of a previous state-of-the-art approach to be used as a performance comparison reference is performed in order to reproduce such results. Then, an evaluation of this thesis’s first contribution (CARD) with respect to power consumption and latency of discovery, is carried out and reported. Moreover, an evaluation of this thesis’s second contribution (ADTP), with respect to the prediction accuracy is reported. Finally, an evaluation of ADTP as a framework for discovery based on the predictions is introduced.

6.1 Introduction

The aim of this evaluation, is to benchmark the performance of this thesis’s contributions against relevant state-of-the-art protocols for discovery. In particular, firstly, the Context Aware Resource Discovery framework has been evaluated against the Resource Aware Data Accumulation framework modified to schedule Disco actions, in order to make the comparison fairer with respect to CARD’s approach. The objective of such an evaluation is to show that CARD can offer lower power consumption and more useful (residual) contact time for communication after discovery in different mobility scenarios. In fact, in IoT scenarios of opportunistic networking, improving IoT devices lifetime as well as providing for longer communication times in scenarios where contacts are rare and short is a desirable property.

In addition, ADTP has been evaluated in the accuracy of the predictions it provides for different mobility scenarios. Such an algorithm has then been ported in NS-3 and a resource scheduler that exploits the prediction framework for discovery has been built. The performance achievable by such a prediction and discovery framework in power consumption and average latency has then been evaluated against CARD showing improvements. Summarizing, the results show that:
• CARD shows improvements with respect of RADA in latency, cumulative residual contact time and energy for the discovery process, especially in scenarios which show more periodicity such as the Deterministic and Multiple Deterministic scenarios. However, in mobility scenarios where such a recurrency is challenged, such as in Gaussian or Real World traces, CARD only shows a limited improvement in latency and cumulative residual contact time. Nonetheless, the energy consumed by CARD in Real World Scenarios is almost one third less than that of RADA.

• ADTP shows improvements as compared to CARD, especially in discovery latency and energy consumption. The prediction based discovery schedule allows obtaining a lower latency in all the scenarios, combined with a lower power consumption, especially but not only with real world traces. The only scenarios in which ADTP consumes more energy are the Gaussian mobility scenarios, in which the randomization also increases the largeness of the low latency phase and thus the energy consumed by ADTP to be able to maintain a latency prevalence over CARD. Similarly, in the STEPS mobility model, ADTP offers a lower latency than CARD but a higher energy usage. This is mainly due to the lower discovery ratio of CARD which fails to find most of the short contacts.

6.2 RADA Validation

In order to validate the Resource Aware Data Accumulation framework implementation, the NS-3 simulator has been setup in order to reproduce the author’s results. In fact, the beaconing strategy that the authors propose in the paper has been adopted, which derives from their previous work \[85\]. In such a strategy, the mobile node is the node in charge of beaconing with a beacon message duration $T_{BD}$ and a beaconing period $T_B$. On the contrary, the static node is awake listening with a duty cycle $\delta$ with an active time $T_{ON} \geq T_B + T_{BD}$, thus selected in order to be sure to recognize at least one beacon when it is woken up. Nonetheless, different from RADA, the Automatic Repeat reQuest with selective retransmission (ARQ) communication phase after discovery is not modelled, since the interest of this evaluation is only in comparing discovery approaches.

In addition, to be the closest possible to the authors’ results, the propagation loss model is modelled as in their work, thus relying on a probabilistic formula which interpolates results from an experiment that has been carried out in their premises \[154\]. In detail, the loss probability is modelled directly in the Channel with the following formula:

$$p(t) = a_2 \cdot \left( t - \frac{c_{max}}{2} \right)^2 + a_1 \cdot \left( t - \frac{c_{max}}{2} \right) + a_0,$$

which holds in the area $0 \leq t \leq c_{max}$ which the mobile node cross at fixed speed and at $D_y = 15m$ of vertical distance from the static node placement.
The simulations have been carried out in order to reproduce the adaptive learning trends of RADA. In particular, a Beaconing Application is installed on a mobile node and, instead, a Learning Application is installed on a static node. Details of all the parameters involved are reported in Table 6.1. As it can be seen from the Table, in fact, the parameters for the RADA Validation Parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Exploration ($\epsilon_{\text{min}}$)</td>
<td>0.02</td>
<td>N/A</td>
</tr>
<tr>
<td>Maximum Exploration ($\epsilon_{\text{max}}$)</td>
<td>0.3</td>
<td>N/A</td>
</tr>
<tr>
<td>Hamming Threshold ($\theta$)</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>Time Domain Duration ($T_D$)</td>
<td>100</td>
<td>s</td>
</tr>
<tr>
<td>Maximum Duty Cycle ($\delta_{\text{max}}$)</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>Price Multiplier ($m_p$)</td>
<td>10</td>
<td>N/A</td>
</tr>
<tr>
<td>Beacon Period ($T_B$)</td>
<td>0.1</td>
<td>s</td>
</tr>
<tr>
<td>Beacon Duration($T_{BD}$)</td>
<td>0.01</td>
<td>s</td>
</tr>
<tr>
<td>$a_0$ at 3.6Km/h</td>
<td>0.133</td>
<td>N/A</td>
</tr>
<tr>
<td>$a_1$ at 3.6Km/h</td>
<td>0</td>
<td>s(^{-1})</td>
</tr>
<tr>
<td>$a_2$ at 3.6Km/h</td>
<td>0.000138</td>
<td>s(^{-2})</td>
</tr>
<tr>
<td>$c_{\text{max}}$ at 3.6Km/h</td>
<td>158.53</td>
<td>s</td>
</tr>
<tr>
<td>$a_0$ at 40Km/h</td>
<td>0.4492</td>
<td>N/A</td>
</tr>
<tr>
<td>$a_1$ at 40Km/h</td>
<td>0</td>
<td>s(^{-1})</td>
</tr>
<tr>
<td>$a_2$ at 40Km/h</td>
<td>0.0077</td>
<td>s(^{-2})</td>
</tr>
<tr>
<td>$c_{\text{max}}$ at 40Km/h</td>
<td>16.915</td>
<td>s</td>
</tr>
<tr>
<td>Radio TX Current</td>
<td>0.0165</td>
<td>A</td>
</tr>
<tr>
<td>Radio RX Current</td>
<td>0.0096</td>
<td>A</td>
</tr>
<tr>
<td>Radio Sleep Current</td>
<td>0.000002</td>
<td>A</td>
</tr>
<tr>
<td>Radio Voltage</td>
<td>3</td>
<td>V</td>
</tr>
<tr>
<td>Radio Range</td>
<td>93</td>
<td>m</td>
</tr>
</tbody>
</table>

CC1000 radio have been used (as in RADA). In addition, the radio range and the error loss model have been modelled according to the parameters reported. Moreover, the beaconing application on the mobile node has the period and duration reported. Finally, the learning application adopts the exploration and the hamming thresholds, the time domain duration and the maximum duty cycle reported.

According to RADA’s results, the Deterministic mobility scenario in which the mobile node moves at 3.6Km/h and 40Km/h entering within communication range of the static node with an inter-contact time of 1800 seconds, has been evaluated. The simulation has been carried out for an extensive time of more than 1000 time domains, corresponding to an equivalent time of roughly 23 days. The number of executions of the various duty cycling tasks over time has been recorded and plotted. In Figure 6.1 it is possible to see the number of tasks execution over time in the case of the mobile node moving at 3.6Km/h. As it can be seen, the very low duty cycle (VLD) always gets the highest number of executions, thus introducing a reduction in the power consumption.
In Figure 6.2 at 40Km/h, the situation shows a trend inversion with respect to the 3.6Km/h, where the high duty cycle task (HDC) gets better reward than the low duty cycle task (LDC) due to a shorter duration of the contacts. In addition, in both figures, it is possible to see the exploration strategy which at the beginning is higher, but reduces itself over time after contacts are made.
In conclusion, RADA’s implementation reproduces the results of the original RADA evaluation, thus allowing us to have a fair comparison basis for the evaluation of this thesis’s first contribution for Context Aware Resource Discovery, which is reported in the following section.

6.3 CARD Performance Evaluation

In order to evaluate this thesis’s first contribution for Context Aware Resource Discovery, RADA’s propagation loss model has been modified and a fading model has been added. This has been made necessary by the requirements to add flexibility with respect to the error loss model presented in the previous section, both in the way of entering the communication range and in the speed to be used in the simulations. In particular, speeds needed to be considered at different values than the ones available from the authors’ experiment measures (see previous section). Moreover, mobile nodes could enter within communication range of the static node from different directions and with different trajectories.

For such reasons a LogDistance propagation loss model has been selected, which assumes an exponential path loss from the sender to the receiver and reflects urban, suburban or indoor scenarios. Moreover, a Nakagami-m Fast Fading model has been used in the simulations in order to account for path reflections in conditions of mobility [155]. In Figure 6.3 it is possible to see the propagation loss over distance for a MICA2 node which has a line of sight maximum propagation distance around 150m. As it can be seen, as the distance increases the transmitted packets have a lower probability to overcome the energy threshold (i.e. -90dBm from Table 5.1).
at the receiver. It is important to note that, the MICA2 node use has been used only because it is used in RADA and to verify that the proposed model is capable of modelling a realistic behaviour. In fact, in the next sections, a more recent TelosB node based model for the radio it is used.

In order to evaluate CARD, different mobility scenarios have been considered as discussed also in the implementation section:

- **Deterministic** scenario consisting of one mobile IoT device entering communication range of a static IoT device, periodically every 1800 seconds (30 minutes), corresponding to the mobility of a robotised controller for collecting data.

- **Multiple Deterministic** scenario consisting of one mobile IoT device entering communication range of a static IoT device, periodically every 1800 seconds, but with a period that increases of 200 seconds every two days up to 5 days and then decreases back to the original period. Intuitively, this corresponds to the mobility of a robotised controller which, autonomously changes its schedule over time.

- **Gaussian** scenario consisting of one mobile IoT device entering communication range of a static IoT device, periodically every 1800 seconds but with a variance of 100 seconds, corresponding to the mobility of a public transportation mean which arrives with an inter-contact time distributed at 99.7% within ± 5 minutes of the 30 minutes inter-contact time.

- **Real World Trace** consisting of Bluetooth logs between a static IoT device and any of all the mobile IoT devices carried by employees in an office environment [150].

<table>
<thead>
<tr>
<th>Table 6.2: CARD Simulation Parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Slot Duration (t_{slot})</td>
</tr>
<tr>
<td>Beacon Duration (t_b)</td>
</tr>
<tr>
<td>Radio TX Current</td>
</tr>
<tr>
<td>Radio RX Current</td>
</tr>
<tr>
<td>Radio Sleep Current</td>
</tr>
<tr>
<td>Radio Voltage</td>
</tr>
<tr>
<td>Radio Range</td>
</tr>
<tr>
<td>Exploration Factor (\epsilon)</td>
</tr>
<tr>
<td>Learning Rate (\alpha)</td>
</tr>
<tr>
<td>Discount Factor (\gamma)</td>
</tr>
</tbody>
</table>

The simulation parameters for CARD (also adopted by RADA) can be found in Table 6.2. In particular, it is possible to find the slots and the beacons durations, used by the underlying Disco schedule, which is also applied to RADA to have a fairer comparison between the protocols. In addition, the more recent CC2420 [148] radio model, with relevant parameters configured, has
been used (used in TelosB nodes). Finally, the learning parameters have been configured to have a high learning rate and a low discount factor as well as a fixed 5% exploration strategy.

For the synthetic traces, in order to cover different contact durations, the mobile nodes have been simulated with the different speeds of 3.6Km/h, 20Km/h and 40Km/h, thus representing contacts durations of, respectively, 200, 36 and 18 seconds. For the real world traces, instead, the contact durations vary over time and have been simulated accordingly. While simulations would allow every IoT device to learn its patterns of interaction with other IoT devices, to keep a fairer comparison with RADA, the learning algorithm is simulated only on the static IoT device, while the mobile IoT devices schedules only fixed low latency sub-actions. For all the simulation scenarios, different algorithms have been evaluated:

- **Oracle**, representing the theoretical optimum discovery algorithm which has perfect knowledge about mobility patterns.
- **Fixed LLSA**, representing the optimal higher bound for discovery latency.
- **Fixed HLSA**, representing the optimal lower bound for discovery latency.
- **CARD**, this thesis’s first contribution featuring Disco actions.
- **RADA**, also featuring Disco actions.

In order to assess the performance of CARD, different metrics have been measured and collected, such as:

- **Discovery Ratio** measured as the percentage (the ratio over the total number) of contacts discovered between IoT devices.
- **Total Cumulative Residual Contact Time** which measures the useful contact time available remaining after discovery between IoT devices.
- **Energy Consumption**, representing the breakdown of energy with respect to the position of the mobile device, such as the total energy spent, the energy spent while outside transmission range and the energy spent while inside transmission range.
- **Average Latency** measured as the mean discovery latency observed between the IoT devices.

A simulation set made of 50 independent runs with 95% confidence interval has been carried out, for an equivalent time of 20 days simulation time for the synthetic traces, and for the necessary duration (about 1 month) for the real world trace. In Figures 6.4, 6.5 and 6.6 it is possible to see the total cumulative residual contact time, the energy breakdown and the average latency for the aforementioned algorithms under the deterministic mobility pattern scenario. However, the discovery ratio is not shown since it is equal to 100% for all the algorithms in all the mobility scenarios.
As for the total cumulative residual contact time, it is possible to see that CARD very closely matches the performance of Fixed LLSA and Oracle, therefore being able to discover IoT devices with low latency. On the contrary, RADA is very close to the performance of Fixed HLSA, since it does not match contacts with low latency, but uses the HLSA since its aim is to reduce energy consumption. CARD in general consumes very low total energy with respect
to all other algorithms, except the Oracle optimum: it consumes about 10% of the energy of
Fixed HLSA and 5% of the energy of RADA.

![Figure 6.6: Average Latency (Deterministic).](image)

This is due to the combination of the HLSA scheduling and of the selective sleeping feature
with which CARD is equipped, which could not be used in RADA, since there would be no
knowledge about when to wakeup after sleeping. In addition, CARD consumes much less when
out of contact with any IoT device: it consumes less than 2% than Fixed HLSA and RADA.
Concerning the average discovery latency, it is possible to see that CARD closely matches Fixed
LLSA for all the speeds, while RADA performs as Fixed HLSA. This is due to the fact that
CARD is capable to learn the pattern of encounters and match it with discovery actions that
contains LLSA when the contact is expected with a high probability. It is important to note
that, for all the algorithms, the average latency reduces proportionally to the contact duration
(i.e. the different speeds) because for shorter contacts to be discovered, the LLSA schedules
a latency bound which is 5% of the minimum contact duration, as reported in Section 3.4.
In addition, the faster the discovery, the lower the energy spent while in contact for CARD,
showing that faster discovery means also lower energy spent for the discovery process. Finally,
since in the deterministic scenario the contact is basically found right away as the action is
scheduled, the energy spent outside contact is almost the same for all speeds (there is negligible
contribution from HLSA).

Another different simulation set has been carried out in order to show what happens when
the periodicity parameter is incorrectly set in CARD thus leading to more than one contact
within an action. A Deterministic scenario at 3.6Km/h in which a mobile IoT device arrives with
an inter-contact time 25% lower than the periodicity parameter has therefore been simulated. Not surprisingly this led to a reduction in the discovery ratio and in the cumulative residual contact time of 25%, therefore of the same quantity, meaning that an action every four presents two contacts, of which only the first is discovered. However, in such a case, the resulting average

Figure 6.7: Total Cumulative Residual Contact Time (Multiple Deterministic).

Figure 6.8: Energy Breakdown (Multiple Deterministic).
latency only slightly increased from 6.52s to 7.53s, with respect to the original Deterministic scenario. Similarly, the total energy consumed only slightly increased from 11.71J to 57.01J, thus remaining 80% less than RADA in the Deterministic case.

The Multiple Deterministic scenario results are shown in Figure 6.7, 6.8 and 6.9. In such a scenario, CARD is challenged by the continuous changes in the mobility patterns.

![Figure 6.9: Average Latency (Multiple Deterministic).](image)

![Figure 6.10: Energy Breakdown (Gaussian).](image)
Concerning the total cumulative residual contact time, while there is a degradation in performances with respect to Fixed LLSA, CARD still outperforms RADA and Fixed HLSA. This is directly related to the continuous changes in the inter-contact times which makes the learning algorithm try to adjust its schedule accordingly.
In addition, such an effect, can be seen as affecting the total energy consumed, which results higher than in the Deterministic case, but still lower than RADA. In fact, by not always discovering the contact with the LLSA action, the energy consumed by CARD when out of contact is increased with respect to the previous scenario. However, such an energy is still lower than the energy consumed by RADA. Finally, similarly as for the other metrics, for the average latency the changes in mobility patterns contribute to a reduction in the discovery latency, which however still remains a 33% less than RADA, with a discovery ratio of still 100%.

The Gaussian mobility pattern leads to an even more continuously changing mobility pattern, with a high range of values and unpredictable behaviour. This puts a very high challenge in both cumulative residual contact time and average latency as can be seen in Figure 6.11 and 6.12. In fact, this lead to a similar behaviour of what happens in the Real World traces where little difference is shown between CARD and RADA, with CARD’s slightly prevalence. In addition, concerning energy, CARD still presents an advantage with respect to RADA as can be seen in Figure 6.10. While for the Gaussian scenario, the similar performances in latency and useful contact time are due to the fundamental unpredictability of arrivals of the mobile IoT device, for the Real World scenario there is a higher predictability, but the inter-contact times present an even higher sparseness across the range of values, as well as a broad range of contact durations.

Concerning the total energy consumed, CARD still prevails on RADA by consuming 63% less than RADA for the Gaussian scenario and 30% less than RADA in the Real World traces. Finally, in order to evaluate the impact of the energy savings, the energy consumed by CARD
and RADA in the Gaussian and Real World Traces scenarios divided by the number of seconds of useful contact time which both algorithms achieve has been measured. Results are reported in Figure 6.13 showing for CARD a reduction of 64% for the Gaussian scenario and of 28% for the Real World Mobility Traces.

6.4 ADTP Predictor Evaluation

In order to evaluate the performance of the prediction algorithm, ADTP has been simulated with different synthetic and real world mobility patterns. Since the objective of this section is to understand how accurate the arrival and departure predictors are, different traces have been simulated in order to evaluate the next arrival and departure prediction error distributions, as computed with Equation 4.26. Furthermore, not only the next arrivals and departures have been evaluated, but also the prediction of the contacts after the next contact (i.e. two steps ahead contact). This has been performed by evaluating the value function’s prediction in order to make a further prediction. Note that, a further multiple steps ahead prediction could be performed by following the states trajectory accordingly, however with decreasing accuracy since no reinforcement of the prediction would take place. Finally, the actual arrival and departure times experienced by the algorithm have been plotted and evaluated against the associated predicted values, thus showing how the system learns over time.

Concerning the synthetic trace, four different mobility scenarios have been considered:

- **Deterministic** where the sequence of arrival times has a fixed 30 min inter-contact time (i.e. time between two consecutive arrivals).

- **Multiple Deterministic**, where the sequence of arrival times has a 30 min inter-contact time which increases every two days of 3 min.

- **Gaussian**, where the sequence of arrival times has a 30 min average inter-contact time but that can assume values normally distributed at 99.7% between 15 min and 45 min.

- **Multiple Gaussian**, where the sequence of arrival times has a 30 min average inter-contact time but that can assume values normally distributed at 99.7% between 15 min and 45 min and that increases every two days of 3 min.

ADTP’s approach has also been evaluated against the real world mobility traces collected during the in-house experiment aforementioned [150], specifically:

- **Bluetooth** trace corresponding to the arrival times representing the interactions of many mobile IoT devices with one static IoT device, in an office environment.

- **P.I.R.** trace corresponding to the presence pattern of a person at a desk in an office environment, measured with a passive infrared (P.I.R.) sensor.
Finally, the real world mobility traces of the Haggle project [151], that are:

- **Intel** trace of Bluetooth sightings between a pair of users (8,4) carrying a small device (iMote) for six days in the Intel Research Cambridge Lab.

- **Cambridge** trace of Bluetooth sightings between a pair of users (7,3) carrying a small device (iMote) for six days in the Computer Lab at the University of Cambridge.

- **Infocom** trace of Bluetooth sightings between a pair of users (19,40) carrying a small device (iMote) for four days during the IEEE Infocom Conference in the Grand Hyatt Miami.

have been simulated. Only one pair has been selected for each trace, with the following criteria: a pair which has seen a number of contacts, which is around mid-way between the maximum number of contacts and zero contacts is selected. Those pairs correspond, in this author’s opinion, to pairs achieving the most “dynamic” traces.
Many pairs are indeed seeing too few contacts to obtain significant results (i.e. 3/4 contacts). Similarly, pairs seeing too many contacts are representing nodes almost all the time within range of each other. Given the conference environment, it is easy to imagine these might have been people sitting near each other. Regarding the simulation time, while the synthetic traces have been simulated for an extensive equivalent time of 20 days, the real world trace simulation time corresponds to the duration of such traces (a month circa for the in-house traces and about a week for the Haggle traces).

As it can be seen in Figure 6.14 the one step ahead predictor for the deterministic arrivals and departures has a prediction error which is distributed at 99% within 1 minute of the actual observed value (i.e. the difference between the actual arrival or departure time and the predicted time is within 1 minute). Such an accuracy reduces to about 95% of the predictions concentrated within 1 minute of the actual arrivals for the two steps ahead predictor, therefore showing a very high accuracy.
6.4 ADTP Predictor Evaluation

This reduction in accuracy from the 100% target is due to the first steps of the prediction algorithms in which predictions that have a higher error are made, due to inaccurate knowledge. Note that, in these and in the next figures, the average error and standard deviation are reported on top. In addition, as shown in the arrival and departure predictions figures, the predictions closely follow the observed arrival and departure times over the entire length of the traces.

In Figure 6.15 the accuracy results for the Multiple Deterministic scenario are shown. As it can be seen, the one step ahead arrival and departure prediction errors are distributed at 96% within 1 minute of the actual arrivals. This reduces to still 91% in the case of the two steps ahead predictor. Evidently, such a reduction from the Deterministic scenario is due to the changes in mobility patterns which require a few iterations of the learning algorithm to adjust the predictions. Similarly to the Deterministic scenario, the arrival and departure predictions closely match the actual observed values over the entire length of the traces.

Figure 6.16 reports the accuracy for the prediction algorithm for the Gaussian distributed mobility scenario. Clearly, both the one step ahead and the two step ahead predictor show a
Evaluation

high error, showing that only 10% of the prediction errors are distributed within 2 minutes. However, as the error interval increases, a higher number of predictions will be relatively close to the actual values.

Figure 6.16: Gaussian Mobility Pattern.
Evidently, by having a standard deviation of 5 minutes for the Gaussian scenario, due to the 68%, 95%, 99.7% rule (i.e three-sigma rule of thumb), it is clearly shown in the one step ahead predictor that already 68% of the predictions have an error within 10 minutes, coherently with the hypothesis of normal distribution. This shows that the predictor is capable of predicting the arrivals in mean with a good guess close to the actual random outcomes. The same trend is confirmed, but with a minor accuracy for the two steps ahead predictor, which has a similar reduction in accuracy with respect to the one step ahead predictor as in the Deterministic and Multiple Deterministic scenarios. Concerning the arrival and departure predictions, the figures show that such predicted values vary in a short range very close to the observed values over the entire length of the traces.

The same trend, which combines effects of Multiple Deterministic and Gaussian scenarios, is shown in Figure 6.17. As it can be seen, the Gaussian trend reduces in accuracy, approximately in the same way as the Deterministic trend reduces to the Multiple Deterministic trend, when the inter-contact time is increased over time.
This causes the predictor to adjust every time a new change in the mobility pattern is experienced by increasing the inter-contact time. As it can be seen from the arrival and departure predictions and observations, the algorithm learns to adapt to such changes in the mobility pattern. Finally, the predictor has been evaluated on real world mobility traces, in order to understand its performance in a realistic environment. In the Bluetooth traces, reported in Figure 6.18, it is shown that more than 50% of the predictions are within 3 minutes of the actual observations. Such a percentage reaches 80% if 5 minutes are considered. As it can be seen by looking at the figures about arrival and departure predictions and observations for arrivals and departures, the predictor is able to closely follow the mobility pattern. In particular, the predictions are challenged only by large abrupt variations in the patterns of arrivals, due to various reasons: i.e. day-night patterns changes, changes between days, user’s choice of changing his habits, etc.
In the P.I.R. traces of Figure 6.19, instead, such an accuracy is not reached, and requires a 10 minutes interval in order to reach at least a 50% percentage.
The 80% percentage is reached only if the interval is enlarged to 30 minutes, which however is relatively large. As for the arrival and departure predictions and actual outcomes, a similar behaviour as the one reported in the previous Bluetooth traces is observed, though reporting a more uniform pattern in these traces. It is also worth noting that the two steps ahead predictor does not significantly degrade the performance of the one step ahead predictor for such real world traces.

In Figure 6.20, it is possible to see the results for the Intel traces of mobility. The accuracy reached by the predictor on such a trace is of the same order of that of the P.I.R. traces. In fact, similarly to the P.I.R. traces, the arrival predictor has 50% of the predictions distributed within 10 minutes of the actual outcomes. However, for the Intel traces, such a predictor reaches a better value of 80% of the predictions within just 15 minutes of the actual values, therefore less than the 30 minutes interval of the P.I.R. traces.
It is interesting to note that, given the relatively low number of contacts (which can be seen by the abscissa of the predictions figures (e) and (f)) and therefore of the amount of data for the algorithm, the arrival and departure error distributions are slightly different from
each other. Furthermore, in the predictions against observations graphs, it is interesting to note the convergence pattern of the algorithm. In particular, within the first 10 steps the predictions are highly inaccurate with possible overshooting or undershooting, while as the learning progresses, they become more accurate. Finally, the two steps ahead arrival and departure errors distribution shows a degradation in the accuracy of the same order of the one of the previous traces.

In Figure 6.21 it is possible to see the results for the Cambridge traces. In such traces, the arrivals predictions are distributed at 50% within just 10 minutes of the actual arrivals, reaching almost 80% within a 25 minutes interval. The errors distributions are therefore in line with all the previous results for the previously analysed real world mobility traces. Concerning the arrival and departure predictions against observations, these traces present very few steep changes in the arrival and departure times. Given the relatively low number of contacts of these traces, it is possible to clearly see the overshoots and undershoots of the predictions. However, as previously explained, ADTP is able to react promptly to such changes by adopting a learning phase in which more knowledge is learned to overcome such changes.
6.4 ADTP Predictor Evaluation

In Figure 6.22 the results for the Infocom Traces are reported. In the error distribution figures, a minor degrade of accuracy is experienced. Only 40% of the predictions are distributed within 10 minutes of the actual observations, with lower figures for the two steps ahead predictor. It is this author’s opinion that this lower accuracy is due mainly to the high presence of abrupt variations within the observed times and partly to the relatively low duration of the traces (four days experiment). This is shown by the predictions and observations figures, where a high number of large variations of arrival times challenges the predictor’s convergence. For example, between the 15th contact and the 35th contact, there are 4 very large variations, thus once every 5 contacts in average. In conclusion, ADTP works efficiently and is able to predict within reasonable figures the next arrivals and departures over various synthetic and real world mobility traces. It is also this author’s opinion that, by incorporating new different features within the framework, the predictor might further improve its accuracy.
6.5 ADTP Discovery Planner Evaluation

While the predictor has a good accuracy in predicting future contacts, an evaluation of the discovery framework built has been carried out in order to compare such a predictor against this thesis’s first contribution CARD, which in turn improves over the previous state-of-the-art algorithm RADA. A simulation under the NS-3 simulator has been performed in different mobility scenarios, similar to the previous ones in which CARD has been evaluated:

- **Deterministic** scenario consisting of one mobile IoT device entering communication range of a static IoT device, periodically every 1800 seconds, corresponding to the mobility of a robotised controller for collecting data.

- **Multiple Deterministic** scenario consisting of one mobile IoT device entering communication range of a static IoT device, periodically every 1800 seconds, but with a period that increases of 180 seconds every two days, corresponding to the mobility of a robotised
controller for collecting data which autonomously changes its schedule.

- *Gaussian* scenario consisting of one mobile IoT device entering communication range of a static IoT device, periodically every 1800 seconds but with a variance of 50 seconds, corresponding to the mobility of a public transportation mean which arrives with an inter-contact time distributed at 99.7% within ± 2.5 minutes.

- *Multiple Gaussian* scenario consisting of one mobile IoT device entering communication range of a static IoT device, periodically every 1800 seconds but with a variance of 50 seconds, corresponding to the mobility of a public transportation mean which arrives with an inter-contact time distributed at 99.7% within ± 2.5 minutes. In addition, such a period increases of 180 seconds over time every two days.

- *Bluetooth Trace* consisting of Bluetooth based logs between a static IoT devices and any of all the mobile IoT devices carried by employees in an office environment [150].


- *Intel Trace* of Bluetooth sightings between users for six days in the Intel Research Cambridge Lab.

- *Cambridge Trace* of Bluetooth sightings between users for six days in the Computer Lab at the University of Cambridge.

- *Infocom Trace* of Bluetooth sightings between users for four days during the IEEE Infocom Conference at the Grand Hyatt Miami.

- *STEPS* mobility model, which follows a truncated power-law distribution as described in the previous chapter. In particular, the Random Waypoint has a speed drawn between 3.6Km/h and 40Km/h, with pause times between 1 and 5 seconds. The attractor power is set to $\alpha = 0.5$ and the temporal preference to $\beta = 0.5$, with a drawing interval between 20 and 30 seconds for the temporal preference. Finally, the grid is a $10 \times 10$ squared area composed by squared 120m by 120m zones.

The simulation parameters for ADTP’s evaluation are the same as the ones in Table 6.2. As for CARD, in order to cover different contact durations, the mobile nodes have been simulated with the synthetic traces for different speeds (3.6Km/h, 20Km/h and 40Km/h) representing contact durations of, respectively, 200, 36 and 18 seconds. Conversely, for the real world traces and the STEPS model, varying contact durations are reported. Different metrics have been collected for both ADTP and CARD:

- *Average Latency* measured as the mean latency observed between IoT devices in percentage with respect to the total contact duration, thus representing the average performance in discovery.
• **Discovery Ratio** measured as the percentage (the ratio over the total number) of contacts discovered between IoT devices.

• **Energy Consumption**, representing the breakdown of energy consumption with respect to the position of the mobile device, such as the energy spent while outside transmission range, the energy spent while inside transmission range, the energy spent inside transmission range before discovery and in misses.

• **Wasted Time Energy Product** which measures the product of the wasted contact time (the sum of the discovery latencies) in discovery and of the energy wasted in discovery when outside contact, thus representing how much energy and communication time is wasted with respect to an optimum Oracle algorithm, which would show a zero value.

A simulation set made of 50 independent runs with 95% confidence interval has been carried out, for an equivalent time of 10 days simulation time for the synthetic traces, and for the necessary durations (about 1 month) for the real world traces.

In Figure 6.23 it is possible to see the results for the Deterministic Scenario simulations. As it can be seen, ADTP presents an average latency more than 30% lower than CARD, while discovering all the contacts with a 100% discovery ratio as CARD. The improvement in latency is due mainly to the fact that, different from previous CARD simulations, the application starting point within a sub-action is randomized. This effect shows that CARD loses performance if the LLSA is not perfectly aligned with the contact starting and ending times, while ADTP, instead is resilient to such an effect. Concerning the energy consumed, as it can be seen, ADTP presents an energy consumed which is as low as 30% of CARD while out of contact in the 3.6Km/h case. The performance improvement with respect to CARD can also be seen in the wasted time energy product metric. ADTP’s wasted time energy product assumes values between 3% and 4% of CARD’s values across all speeds.
Finally, while the energy for misses is negligible, the energy spent during the discovery shows a higher value for ADTP, confirming ADTP’s attitude to concentrate its resources during the phase of approaching (entering communication range) a potential contact opportunity.

In Figure 6.24, it is possible to see the results for the Multiple Deterministic scenario. Similarly to the Deterministic scenario, ADTP has an advantage with respect to CARD in average latency while discovering with a 100% discovery ratio. However, only a 17% lower latency is reported, due to the challenged scenario of mobility introduced by the varying inter-contact time. Concerning the power consumption while out of contact, a similar reduction as in the Deterministic scenario is observed for ADTP, to a slightly higher value of 40% of CARD’s consumption, except that for the 40Km/h case, where the node presents a shorter contact with respect to the inter-contact time. In such a case, in fact, the contact duration at 40Km/h is only 18 seconds, while at 3.6Km/h is 200 seconds. This means that an increase in the inter-contact time of 3 minutes (180 seconds) will change significantly the predictor’s outcome for the 40Km/h case rather than for the 3.6Km/h case.
The change in the mobility patterns, therefore triggers a transient period in which more energy is spent because the mean square prediction error is higher and therefore the second phase duration increases temporarily in order to learn the new pattern. Concerning the wasted...
time energy product, this increase in energy reflects into a higher value for the 40Km/h, which however is still 18% of CARD’s value. Finally, the energy spent in misses and in discovery show the same pattern as for the Deterministic scenario.

In Figure 6.25 it is possible to find the results for the Gaussian Mobility scenario. As it can be seen, ADTP still outperforms CARD in average latency proportionally in the same manner for all speeds, while keeping a discovery ratio very close to 100%. However, while at lower speeds ADTP consumes less energy while out of contact than CARD, at higher speeds, in order to achieve a lower latency, which CARD could not achieve, the algorithm “dynamically” spends more energy by widening its second phase. In fact, due to shorter contacts at higher speeds, it becomes more difficult for the predictor to match a short contact in a wide interval thus leading to a higher mean squared prediction error, which leads to a longer second phase. The same results are also confirmed by the wasted time energy product metric. Finally, concerning the energy spent during misses and in discovery, the trends are the same of those of previous scenarios.
In Figure 6.26 showing results for the Multiple Gaussian scenario, it is possible to see the same trends of the Gaussian scenario.
This confirms the intuition that in random patterns of arrivals, it is indeed difficult to predict the arrival as the next draw. ADTP can in fact only predict the next arrival “on average” based on the sequence of learned values.

As for the real world traces, the results obtained for the Bluetooth Traces are reported in Figure 6.27. Concerning the average latency, ADTP achieves a lower value of 6.7% of the contact with respect to 14.47% of the contact of CARD, while still keeping a 100% discovery ratio. This is achieved in combination with a power consumption when out of contact for ADTP which is 7 times less than CARD. This is also reflected into a very low wasted time energy product metric for ADTP which is 7% of CARD’s value. Finally, results in line with previous scenarios for the other energies for misses and discovery are reported.
Figure 6.27: Bluetooth Trace Mobility Scenario Results.

In Figure 6.28 as reflected in the accuracy figures of the previous section, for the P.I.R. Traces there is a decrease in performance with respect to Bluetooth traces.
However, ADTP still maintains an edge over CARD for the average latency while keeping a 100% discovery ratio for both approaches. As for the power consumption while out of contact, ADTP shows a 2.5 times less energy than CARD, reflected also into the wasted time energy product metric for ADTP which is roughly 46% of CARD’s value. Concerning the energy spent in misses, as it is possible to see from the figures, both algorithms show a negligible contribute. Similarly to previous results, the energy spent during discovery is also higher for ADTP, meaning that such an algorithm is capable to concentrate its effort during discovery.

In Figure 6.29, it is possible to see the results for the Intel traces of mobility. As it can be seen, ADTP maintains a performance edge over CARD for the average latency by roughly a 15% value. Concerning the discovery ratio, both ADTP and CARD are capable to discover all the contacts with a 100% discovery ratio, but ADTP is able to achieve a lower energy while out of contact, which is 5 times less than CARD.
Evaluation

(a) Average Latency

(b) Discovery Ratio

(c) Energy Out of Contact

(d) Energy for Discovery

(e) Energy for Misses

(f) Wasted Time Energy Product

Figure 6.29: Intel Trace Mobility Scenario Results.

Such trends are confirmed also by the wasted time energy product of ADTP which is the 3.5% of CARD’s. Finally, while the energy for misses is zero, the energy spent in discovery shows the usual trend.
In Figure 6.30 it is possible to see the results for the Cambridge traces. ADTP presents an average latency which is 5% less than the CARD’s latency while keeping a discovery ratio very near to 100% for both approaches.

Figure 6.30: Cambridge Trace Mobility Scenario Results.
However, ADTP consumes 40% of the energy of CARD while out of contact thus, globally presenting a wasted time energy product metric for ADTP which is 36% of CARD’s value. Both ADTP and CARD present also a negligible energy spent during misses, while showing the usual trend for the energy spent during discovery.

In Figure 6.31 it is possible to see the results for the Infocom traces. ADTP presents an average latency which is 9% lower than CARD’s latency. In addition, both approaches keep a discovery ratio which is around 100%, thus discovering all the opportunities. ADTP also presents an energy outside contacts which is roughly 46% the energy spent by CARD. Concerning the wasted time energy product, ADTP shows a value which is 80% less than CARD’s value, while maintaining a zero energy spent for misses and the same trend for the energy spent during discovery.

Results for the STEPS mobility model are reported in Figure 6.32. Contrarily to ADTP, CARD shows a very low discovery ratio of only 21.49% with respect to the 82.53% of ADTP.

(a) Average Latency
(b) Discovery Ratio
(c) Energy Out of Contact
(d) Energy for Discovery
This is due to the characteristics of STEPS mobility model, which models a truncated power law for the inter-contact times.
In fact, in such a model very short contacts (as short as 1 second) with very low inter-contact times are highly probable, thus posing a challenge on CARD’s action duration scheduling. Since ADTP is not influenced by such parameters, it achieves a higher discovery ratio. The average latency is also very low in ADTP, corresponding to roughly 10% less of CARD’s average latency. However, in order to achieve such an advantage, ADTP needs to schedule much more energy than CARD, and this is coherent with the fact that CARD will sleep most of the time, but without finding almost 80% of the contacts. This also translates into a higher wasted time energy product for ADTP, which however is also influenced by CARD’s lower discovery ratio.

6.6 Conclusions

In conclusion, ADTP and CARD have been evaluated and proven in their advantage with respect to current state-of-the-art in neighbour discovery. In particular, an evaluation of CARD has been performed showing improvements with respect to RADA in terms of power consumption and latency of discovered contacts. This allows IoT applications that want to use CARD as their framework to adapt the resources tailored to their requirements and still guarantee optimal performance. Concerning ADTP, instead, a new prediction algorithm has been developed in order to be used to predict the next arrivals with a good accuracy and that could be exploited in other tasks, rather than for discovery. For example, applications might exploit the predictions for scheduling data collection in an optimized fashion. Finally, ADTP has been incorporated in a discovery framework which showed that it is capable of providing an optimized discovery in power consumption and latency. In particular, ADTP’s framework is capable of providing a latency efficient discovery, which schedules as many resources as needed to guarantee such a discovery, but still guaranteeing lower power consumption than CARD in many scenarios.
It is this author’s opinion that, depending on the scenarios considered, either ADTP or CARD might be used. In particular, when the scenarios show high randomness, corresponding, i.e. to a Gaussian mobility scenario or to the STEPS mobility model, CARD might be used due to its energy efficient superior behaviour to ADTP. However, in scenarios of Opportunistic Networking, applications might require guarantee of data delivery between IoT devices. In such applications, the ADTP’s advantage of being able to predict the actual arrival and departure times of future contact opportunities opens up the possibility not only to optimize the latency of the discovery process thus optimizing the useful time for communication, but also the possibility to implement on top of that knowledge a communication planner for scheduling the transmissions over time. For example, by exploiting the predicted contact duration, a scheduler might decide which backlogs of data are better suited to be transmitted within the next contact duration, or even decide whether to exploit such opportunities in lieu of future more favourable contacts. As it will be shown in the next concluding section, this is the approach chosen for future work.
Chapter 7

Conclusions

This chapter draws conclusions about this thesis and outlines future research plans on discovery and communication in IoT scenarios for opportunistic networking. It is this author’s opinion that knowledge about mobility patterns could not only benefit the discovery process, but also the communication, thus paving the way for a more efficient opportunistic networking.

7.1 Closing Remarks

In this thesis, the problem of how to acquire knowledge about the availability of devices in the neighbourhood in a distributed fashion has been tackled. Such knowledge has been used to optimize the discovery process in IoT scenarios of opportunistic networking. Reinforcement Learning techniques have been used to learn patterns of encounters between static and mobile IoT devices and to schedule resources in an efficient way. In such settings, while more resources are scheduled when other IoT devices are learned to be present within communication range with a high probability, conversely less resources are scheduled when other IoT devices are deemed to be within range with low probability. This helps in reducing power consumption and improving lifetime of IoT devices, thus avoiding energy wastage by unnecessary probing at times when other IoT devices are not present. It is important to note that, in many scenarios, energy is a major constraint that pose a threat on the actual viability of applications (e.g. wildlife monitoring scenarios).

Furthermore, the optimization of the discovery latency when contacts are present with high probability has allowed for longer communication times, which are essential in IoT scenarios in which short and rare contacts between devices might be the only mean to collect data and communicate between devices. By being able to know with a high probability when and for how long contacts will occur in future opens up potentialities for optimizing communication. For example, shorter contacts might be discarded in favour of longer contacts, or backlogs of data of different sizes might be distributed among contacts of varying durations, thus optimizing
This thesis’s first contribution for Context Aware Resource Discovery has in fact allowed learning contact patterns over time and adapting resources in order to save energy when contacts are not expected, while, at the same time, providing a latency optimized discovery when nodes are learned to be in contact. This is achieved by modelling the environment as a Markov Decision Process in which the states are equivalent to the beacon reception patterns over time and the actions are composed of energy and latency jointly optimized sequences. Thanks to a reward function driven by latency, a Q-Learning based algorithm has allowed CARD to learn a policy which optimized energy and latency. In addition, making CARD application driven has allowed optimizing the discovery process subject to application requirements through parameter customization.

This thesis also answers the problem of finding when an opportunistic contact will manifest itself in the future and for how long such a contact will last. This has allowed planning the resource allocation for the discovery process. This thesis’s second contribution, consisting of an Arrival and Departure Time Prediction framework, has in fact allowed predicting the next future arrival and departure times between interacting IoT devices, thus inferring when and for how long future contacts will manifest and last. Based on such a prediction algorithm, a discovery framework has been built and made capable of optimizing power consumption in the time window between the end of the contact and the future predicted arrival. In addition, a customized low latency discovery has been performed when other IoT devices are predicted to be in range with a high probability, based also on an accuracy estimate that the predictor offers. Moreover, a mechanism capable of recognizing abrupt changes in the mobility patterns has been provided in order to trigger learning phases in which more information is incorporated within the predictor.

In conclusion, this thesis provides learning frameworks for mobility patterns aimed at discovery, which can be generally and widely adopted by heterogeneous IoT devices. This is possible because of the reinforcement learning temporal difference methods, which require no training sessions and a limited volume of data (a few iterations) to operate correctly as well as adapting to changing conditions, thanks to their online and trial-and-error nature. In addition, the adoption of temporal overlap based asynchronous discovery protocols gives the framework the possibility to apply such protocols to a wide range of IoT devices, because they do not need any synchronization.

### 7.2 Future Work

One of the benefits of ADTP is the possibility of planning resources, which not only allows to optimize discovery, but also opens up the possibility to plan communication by allowing decisions whether contacts should be exploited or discarded in lieu of subsequent contacts. In fact, depending on application requirements, contacts might be exploited for communication or
skipped in favour of future contacts deemed more appropriate for communication: i.e. shorter contacts could be discarded in favour of longer contacts. This is possible, since ADTP is able to infer contact durations as a difference between the predicted departures and the predicted arrivals. Potentially, this means that, as future extensions of ADTP, short unmeaningful contacts could be discarded in lieu of future more favourable contacts. In addition, by having knowledge about future contact durations, data might be more efficiently matched to different encounters of varying duration. For example, a backlog of data composed by files of different lengths might be intelligently allocated into different future contacts, thus optimizing the data delivery with respect to a “greedy” scheduler.

Future plans entail different perspectives, not limited to communication, but also including improvements to the prediction framework. Since ADTP’s prediction framework employs only arrival and departure times as features for the mobility patterns in order to predict future arrivals and departures, evaluating different features might enhance the accuracy in the predictions. A promising direction could be to introduce additional contextual knowledge in the mobility patterns. Most of the mobility is in fact due to people moving around, whose actions are a consequence of their preferences. For example, every day we carry our smartphones around while going to work in the usual places, along the usual roads and encountering our friends and the people who we work with.

The number of visits or the amount of time spent in certain locations, in combination with temporal features might, for example, help to decide where and when the next contact will be. These features could also be inferred through audio sensors, instead of relying on radios, if benefits in energy are introduced, thus sampling how noisy the environments are and learn if there exists a pattern. Moreover, by incorporating knowledge about one person’s ranking of preferred locations, together with information about friendship between people or community membership (i.e. co-workers or people who share locations such as relatives) could help to reach an improved prediction accuracy.

Furthermore, social behaviour or tagging of the most meaningful locations (i.e work location), also from a device carrier point of view, can help to more precisely infer the patterns of interactions between IoT devices. Finally, employing non-linear value function approximators for Reinforcement Learning, such as Artificial Neural Networks could help to devise more efficient learners. Recently, deep ANN architectures, though more difficult to train, have been used in many applications thanks to their property of autonomously constructing features representation from input layers without requiring the hand-crafting of such features.

Exploiting knowledge about contact arrivals in a distributed but cooperative fashion for opportunistic IoT networks could also be considered in order to share across devices knowledge about future arrivals predictions (i.e. value function parameters), in order to defined “safer” delivery routes for data in advance. This means that nodes that are more mobile than others (seeing more contacts or seeing contacts earlier than others and for more time) could be used to relay data.
Finally, an evaluation on real world IoT devices equipped with multiple radios, such as recent smartphones or recent IoT platforms could be carried out as proof of concept.

7.3 Publication List

During my PhD, I contributed to the following publications:

- *Neighbour Discovery for Opportunistic Networking in Internet of Things Scenarios: A Survey*. IEEE Access, Special Section on Artificial Intelligence Enabled Networking.


- FP7 ICore Project Deliverable *D3.1 Virtual Object Requirements and Dependencies*.

- FP7 ICore Project Deliverable *D3.2 Real Object Awareness and Association*.

- FP7 ICore Project Deliverable *D3.3 Virtual Object Concept Definition and Design Principles*.

- FP7 ICore Project Deliverable *D3.4 Virtual Object proof of concept*. 
References


BIBLIOGRAPHY


