Developing robust Iot to monitor smart spaces at scale

Francoise Sailhan

université PARIS-SACLAY

le cnam

Mémoire présentée en vue de l'obtention de
l'habilitation à diriger des recherches
Université Paris-Saclay

❧

Discipline : Informatique

Developing Robust IoT to Monitor Smart Spaces at Scale

Habilitation à diriger la recherche

Par : **Françoise Sailhan**

Sous la direction de ,

Membres du jury:

Rapporteur: Julien Bourgeois, Prof. des Universités, Univ. de Franche Comté,

Rapporteure: Siobhán Clarke, Prof. des Universités, Trinity College,

Rapporteure: Nathalie Mitton, Directrice de Recherche, Inria-Lille,

Marraine: Valérie Issarny, Directrice de Recherche, Inria-Paris,

Examinatrice: Nalini Venkatasubramanian, University of California,

Présidente: Karine Zeitouni, Prof. des Universités, Université Paris-Saclay,

Date de soutenance : 14 janvier 2021

Contents

Chapter 1

Introduction

1.1 Context and Related Opportunities

The Internet of Things (IoT) has emerged as a distributed, self-healing and large-scale architecture composed of a variety of objects that monitor and affect their environment.This combination of sensing and actuation capabilities holds great promise as the enabler of a wide spectrum of applications, *e.g.*, smart space (*e.g.*, building surveillance and ecology-friendly control), emergency response and environmental monitoring. Instrumented with sensors, RFIDs and actuators, smart spaces (including the residents, employees) provide valuable information that serve supporting *e.g.*, access control [66, 26], inform rescuers in case of emergency [137, 12, 56], provide assisted/enhanced living [179], *ect.*. In the area of environmental monitoring for example, large cities have made progress, in recent years, in monitoring and assessing the biological and ecological impacts of pollution to respond appropriately, from the political to the individual level. Cities have traditionally relied on sensing stations that gather observations (*e.g.*, atmospheric conditions, noise levels, temperatures). Recently cities have also begun taking advantage of crowdsensing movement, wherein users (citizens, groups, communities) rely on the small and low-cost sensors embedded in –or connected to– their smartphones so as to help identify environmental problems. IoT-based applications build on a communication and computing infrastructure spanning different scales, from smartphones embedding or connecting to sensors (*a.k.a.* detectors) and actuators (*e.g.*, electromechanical and/or software-based devices), enabling them to capture, measure, react and communicate with their environment, all the way to large-scale networks, capable of disseminating critical information from event sources to multiple recipients. At the network edge, some wireless networks (*e.g.*, Device-to-Device and wireless sensor networks) with no specific underlying infrastructure in place, connect diverse Things hosting sensors and actuators (*e.g.*, mobile phones, vehicles, smart clothing, motes). Interestingly, these networks mirror *ad hoc* networks — some of the nodes are battery powered and communicate via wireless connections, possibly involving intermediate/proxy-nodes and using multi-hop routing.

So far many specific, distributed, energy-/memory-/computationally-aware protocols, systems and applications have been proposed to enhance the collection and processing of the increasingly massive amount of available information in the IoT. However as the range of applications extends to the fields of industrial and mission-critical systems, additional assurance requirements related to accuracy, reliability and security must be considered. Accomplishing the challenging task of Developing Robust IoT therefore necessitates: (i) supporting accurate and resource-efficient sensing and actuation capabilities and (ii) providing efficient and reliable communication among Things/users across heterogeneous and volatile networks from the edge up to the cloud.

1.2 Challenges of designing and deploying an IoT system

While expectations are high for the wide applicability of IoT systems, the need to monitor phenomena at unprecedented scale is far from being a trivial concern especially considering that IoT applications must remain dependable under any circumstance (*e.g.*, despite disruptions, harsh conditions or even attacks). IoT system applications are subject to a variety of faults/threats, spanning hardware, software and networking layers. Given that the IoT ecosystem is information-centric (*i.e.*, Things generate a torrent of information delivered to many recipients that in turn must filter, customise, and process it on the fly), we are mostly concerned with the two following classes of impairments:

- Communication faults - Any intentional or accidental attempt to disrupt the flow of information seriously jeopardises IoT systems. IoT systems are actually very difficult to insulate against communication faults due to the openness of the wireless communication medium, coupled with the cooperative nature of the network, which render protocols and related applications easy to compromise. In this regards, detecting any misuse or anomaly (*i.e.*, deviations from the normal behaviour) that threatens the networking structure – from edge to core networks – is of critical importance. In this regard we began (i) investigating the detection of misuses and anomalies. Our attention was specifically directed at detecting attempts to disorganise the routing or disturb the data flow relaying. In addition we aimed at detecting anomalies corresponding to certain types of misbehaviour (*e.g.*, unusual and potentially unauthorised access to some data, unexpected presence in a smart building).

- Information-related faults - The collection of data of sufficient quality is a major challenge facing the IoT. Many sensors in use are low cost, mass produced and are made of off-the-shelf components and their relatively low accuracy challenges the relevance and accuracy of the collected sensory data. In addition, the exposure of the sensors to natural perturbations leads to questioning the accuracy of the observations. Correcting inadequate observations in order to derive proper conclusions is therefore of prime importance. Meanwhile, knowledge of the sensors' context is vitally important in order to discover whether the sensing devices are in a position that enables sensing, instead of interfering with it. In response to this challenge we propose to: (i) calibrate the inexpensive (and often inaccurate) sensors and (ii) fuse the information provided by multiple sensors so as to supply more accurate/reliable observations (as opposed to trying to improve the reliability of the individual devices involved).

While IoT enables environmental monitoring at an unprecedented scale (leveraging calibrated sensors and information fusion), it also involves significant communication, computation, and, therefore, financial costs due to the reliance on cloud infrastructures for processing the spatio-temporal data. The situation is actually worsen due to the fact that IoT systems require (i) a low-level control loop to support timely and situational-aware sensing and actuation and (ii) a higher-level control loop to infer a physical phenomenon as a whole, *i.e.*, at macro scale and across time. As an illustration, knowledge of the level of ambient pollution (*e.g.*, noise level, air quality) within a particular area helps citizens measure their own exposure to pollution. An extended control loop is also needed so that citizens, authorities and decision-makers can design adequate policies. As an alternative to the centralised gathering and analysis of crowdsensing observations through such a high-level control loop, we introduce a crowdsensing system that functions on a collaborative model, operating primarily at the very edge, *i.e.*, empowering smartphones so as to enhance the quality of the data transferred to the cloud while

reducing the related communication cost and resource consumption. We adopt an information-centric paradigm in which the information drives decision-making. To achieve this we build upon a publish/subscribe system that supports a loosely-coupled communication towards/from Things/users across heterogeneous and volatile networks in which data flows are governed by both user interests and data content.

1.3 Outline

This document brings together a large portion of the research I have been conducting over a period that spans 2005-2020. After completing my PhD in July 2005 at Inria on the subject of service discovery in mobile *ad hoc* networks, my work has covered several topics at the heart of what is today the (mobile) IoT, with an emphasis on the effective gathering of accurate sensor data. However, I do not detail all of my work thus far; in particular, my industrial research on network management [145, 15, 14, 48, 139, 142, 59, 80] and my work on sentiment analysis [104, 103, 89] are deliberately omitted. This manuscript is instead specifically focused on my research contributions to the design of robust distributed IoT systems supporting the accurate monitoring of urban-scale phenomena, leveraging sensors and actuators embedded in or connected to devices (*e.g.*, smartphones or motes). The focus of this research is the support of robust routing, observation gathering and processing and overcoming misbehaviour or anomalies, while involving end-users as little as possible.

The core of the manuscript is structured around the following chapters, each covering a specific theme of the aforementioned research.

Chapter 2 – Into the Wild: Resource-Efficient Intrusion Detection for Secure IoT

- **Publications**: [99, 6, 8, 7, 141, 146, 66, 22, 138, 126].
- **Supervision**: Mouhannad Alattar (PhD), Khaled Gari (Master student), Christophe Pitrey (Master student).
- **Project on the topic**: FP7 Securinet.

The starting point of my research concerns the detection of intrusions that threaten an IoT system. This usually consists of diverse networks, ranging from *ad hoc* networks dealing with mobile devices, highly constrained RFID (Radio Frequency Identification) systems and also the core networks. In order to protect the networks against malicious activities, we explore two complementary approaches that relate to the detection of misuses and anomalies. We introduce a misuse detection system [7, 146, 127], which is intended to detect an attack based on a predefined attack signature (*i.e.*, a series of operations threatening security). Our anomaly-based detection [66, 22] consists in establishing the "normal"[1] behaviour of the system to be protected and identifies an anomaly as a deviation between a given observation(s) and the pre-established normal behaviour. With our misuse detector, our primary objective is to model the general form of attacks and identify the related attack signatures. Our signature-based intrusion detector distinguishes itself with respect to the state of the art because it uses logs and does not inspect the network traffic. Relevant logs are categorised and exchanged according to their degree of importance. An additional issue is the cost associated with the acquisition and processing of the logs. Our approach [99] to reducing related resource consumption is based on probabilistically questioning neighbours and controlling the flow of logs gathered through the

1. Alternatively, the "abnormal" behaviour of the system can be expressed.

use of statistical parameters. We further establish the trustworthiness [6] of the interrogated node(s) and filter incorrect logs supplied by misbehaving ones. We then design an anomaly detection system [66] that provides early warnings about anything out of the ordinary (*e.g.*, zero-day attacks). Leveraging advanced machine learning techniques, our detector automatically categorises (normal) activities without supervision.

Chapter 3 – Enhancing the Observation Quality through Calibration

- **Publications**: [148, 179].
- **Supervision on the topic**: O. Tavares-Nascimento (Master student).
- **Project**: MINES project in collaboration with UC Irvine.

Environmental monitoring represents a class of applications with an unprecedented benefit for the scientific community and society as well. Outfitting our physical space and people with low cost sensors can enable long-term data collection at scale, but poses the fundamental challenge of gathering high quality data from low-cost sensing devices. It is well-known that sensors are prone to faults, bias and sometimes drifts, which is unfortunate, given that the ability to report accurate and reliable data motivates the use of sensors. A typical approach to enhance the quality of the data and hence fully exploit the potential of sensors, is to calibrate sensors. Traditional calibration processes are carried in an environmentally controlled environment, *i.e.*, typically using a stimuli and in a control chamber. But, calibration in the field is essential to ensure a proper operation of the sensing device, as aging, external conditions (such as solar radiation) and other factors (e.g., activity of the end user) affect sensor's measurements over time. To this end, we propose to carefully plan the sensors calibration and to send mobile units (e.g., trained personnel) equipped with high-quality (more expensive) and freshly-calibrated reference sensors to carry out calibration in the field. The proposed calibration solution is particularly well suited for the calibration of IoT infrastructures as it significantly reduces the cost associated to the upkeep of the sensors in place. Going one step further, we introduce an approach supporting the calibration of mobile crowdsensors without requiring the user involvement because they unlikely have the necessary expertise to do so in an accurate way. We thus introduce a collaborative and automatic calibration among nearby sensing smartphones while capitalising on the calibrated IoT infrastructure, whose sensors provide a bootstrapping calibration to the mobile crowdsensors that are passing by. This leads to the introduction of a novel macro-calibration problem where numerous devices calibrate without involving the end-users.

Chapter 4 – On the Move Again: Leveraging Crowdsensors to Support Robust and Context-Aware Sensing

- **Publications**: [52, 53, 51, 54].
- **Supervision on the topic**: Yifan Du (PhD).

The recent proliferation of human-carried mobile devices has given rise to a new class of IoT applications, called mobile crowdsensing, which aims at outsourcing the collection of sensory data to participating users. However the accuracy of the sensory data depends highly on the expertise of the participants and their behaviour, which should not interfere with the sensing of physical phenomena. At the same time, the involvement of many (possibly incentivised) participants is financially challenging and demanding from the cloud perspective, while it does not necessarily lead to a proportional increase of accuracy and spatio-temporal coverage. A

crowdsensing platform has to therefore correct, properly filter, and interpolate the contributed information as much as possible, so as to deal with missing values and cancel out any possible errors and omissions arising from faulty devices or inexperienced participants. In order to address these issues, we introduce a context-aware [52, 53] and collaborative crowdsensing approach that operates at the edge, where co-located crowd-sensors, operating in the same context, group together [53] to share the workload in a cost- and quality-effective manner. In particular, the most adequate crowdsensors are assigned the crowdsensing tasks, according to the nodes' abilities (for instance a smartphone located in a pocket cannot adequately sense the surrounding sound level). We then jointly distribute the processing of contributed data over the crowdsensors: the collected data is aggregated and the partially observed physical phenomenon is interpolated collaboratively, so as to offload the cloud and overcome any spatio-temporal sparsity.

Chapter 5 – Information-Centric Networking within the IoT

- **Publications**: [82, 140, 147, 82]
- **Supervision on the topic**: Zaid Anwer, Donnacha Nylan (Master students).
- **Project**: European Madeira project, Ericsson Omega project.

Current IoT applications typically sense physical phenomena locally, and usually outsource their authority over information to the cloud. The cloud infrastructure further distributes sensory and control data to end-users and external services. Nonetheless, highly-constrained IoT devices usually work more efficiently in a loosely-coupled environment without maintaining end-to-end connectivity. We therefore adopt an information-centric networking approach [82] and introduce a purposely-built content-based publish-subscribe system. This supports a high demand by decoupling (in time and space) content consumers from data producers. As a result, data producers, *e.g.*, (crowd)sensors, may leverage this loose coupling to reduce their duty-cycle and increase their battery life. Subscribers identify interesting content topics by stipulating the attribute(s) of the content relevant to them. Such a content-based publish/subscribe system improves the expressiveness of subscriptions, hence reducing the dissemination of irrelevant notifications.On the downside, the subscription process comes with a sophisticated and hence resource-consuming filtering and hop-wise forwarding. We improve scalability and expressiveness, although they are two conflicting goals, by supporting a lightweight filtering [147] and distributed notification [143]. In particular, we propose a compact approximation of the subscriptions [140] that speeds up the subscription lookup. We then introduce a self-organising, cluster-based hierarchical structure, which ensures a strict control on the underlying structure and enables the aggregation and correlation of notifications.

Chapter 2

Into the Wild - Intrusion Detection

2.1 Introduction

The practical realisation of IoT involves the design and development of various subsystems (*e.g.*, platforms, protocols and technologies) that identify, sense, actuate, communicate and process at different levels of sophistication. That said, IoT subsystems development is mainly profit-driven and constrained by a short time-to-market, which has historically led – and still leads – manufacturers to overlook security considerations. As a matter of fact, poorly designed IoT devices open the door to adversaries, who often exploit *Things* with little or no effort. Moreover, Things are easily accessible as they usually remain outside of properly protected and compartmentalised networks. The seriousness of the situation is well illustrated by an attack launched by IoT-specific malware called Mirai [2]. Dyn, a leading DNS provider in the United States, has been the target of one of the largest Denial of Service attacks ever launched on a myriad of vulnerable Things. The attack hampered name resolution and caused considerable collateral damage as users were unable to access websites. Given the Internet-wide deployment of IoT, any malicious manipulation may have a profound effect on the resilience of the entire Internet. It therefore becomes vitally important to track events occurring in today's IoT networks and to analyse them for potential signs of security breaches. Once an attack is detected, proper counter-measure mechanisms must be considered to prevent the attacker from causing widespread damage. One straightforward approach to securing the myriad Things is to redesign them, incorporating security agents within their structure. Such an approach would be neither affordable nor would it scale up to the massive amount of Things. This brings us to introduce an Intrusion Detection System (IDS) capable of detecting security threats within IoT environments without any design alterations.

In the course of this chapter we focus on Intrusion Detection, with the aim of discovering whether or not the IoT ecosystem is functioning normally. Our scope covers the search for: (i) *misuses*, which correspond to some specific attacks that are already known and (ii) *anomalies*, which occur when an intruder's behaviour does not conform to the expected or legitimate behaviour. We introduce a *misuse detector* that exploits current knowledge about existing attacks in order to look for evidence/symptoms that signal an attack development. An *anomaly detector* further builds a reference model describing the usual behaviour of a given system and subsequently searches for any noticeable deviation from this model. Whilst essential to avert and control the risks associated with attacks, the general detection of misuses and anomalies that threaten the IoT at large is undeniably an ambitious project. Indeed, the wide range of attacks [163, 38] target different levels, covering hardware, networks, systems, and applications. It is moreover necessary to consider and handle: (i) The heterogeneity of Things, from highly resource-constrained Things (*e.g.*, RFID tags and readers) to personal laptops and (ii) the various underlying organisations, from fully cooperative systems to individual Things.

To this end we cover two case studies, ranging from intrusion detection in Mobile Ad hoc NETwork (MANET) to anomaly detection in RFID-based access control systems. In both cases we consider the nature of the monitored system (network/computer/object and supplied

service) to determine intrinsic vulnerabilities and their corresponding attacks.

We begin with the problem of intrusion detection for the routing protocol in MANETs (§2.2). Our choice is motivated by the fact that a routing protocol supplies a critical network service and is therefore a prime target. Unlike existing systems that monitor packets passing through host [5, 168], our signature-based intrusion detector analyses logs to identify patterns of misuse. Thus, this approach does not necessitate modifying the implementation of the routing protocol, nor does it require inspection of the traffic. Our detector relies on the attack signatures that we pre-established. To this end, we have surveyed and modeled the attacks in a way that reflects the dependencies between their constituting tasks. In particular, we focused our attention on the intrusions threatening the Olsr routing protocol [39]. Based on the attack signatures, our detector discovers the attacks threatening the Olsr protocol. Whilst essential, intrusion detection faces two problems. First is the minimisation of computation overload and bandwidth usage associated with the investigation and intrusion detection processes. Second, the intrusion detection process involves open and spontaneous cooperation with other devices, which are generally unknown or barely known. It is very likely that malicious nodes intend to take advantage of this lack of mutual knowledge in order to disrupt the detection process. It is therefore indispensable to develop a cooperative intrusion detection system that: (i) regulates the amount of remote logs/evidences that are collected and (ii) assesses the trustworthiness of the devices involved in intrusion detection and weighs the contributed logs/evidences accordingly. Our strategy to reduce resource consumption Things on the development of a lightweight and distributed intrusion detection system that analyses logs as close as possible to the device that generated them. The intrusion detection system selects a subset of nearby nodes that are randomly and uniformly questioned. By leveraging the statistical approach (and in particular the confidence interval and confidence level), we reduce the resource consumption and regulate the gathering of evidence without compromising detection reliability. We then incorporate a mechanism for assessing trust and supporting informed decision making. Before considering evidences from another device, a node assesses the device's trustworthiness, which reflects the node's evaluation of its past activities and future intentions. Establishing a relationship of trust is a challenge, not only because it must be insusceptible to deception, but it must also be able to bypass attackers who adapt their behaviour, i.e., behave well and then misbehave in order to abuse the system. To this end we introduce an entropy-based trust system, which aims to objectively express an opinion on a device based on its actions and its recommendations from other nodes. This system capitalises on the notion of entropy, which allows it to establish the similarity between a personal opinion and the received recommendations while determining the extent to which a node's behaviour is legitimate.

Next, we explore a complementary approach that detects anomalies (§ 2.3), considering a case study in which individuals are monitored using RFID tags. Our objective is to depict the normal behaviour of the system and classify any deviation as an attack. We address the problem of detecting anomalies in an RFID system, using unsupervised algorithm. Rather than relying on a simple statistical method to detect any deviation from the normal activity, we select the Kohonen's self-organising maps [92] as an advanced neural network architecture that permits the user's profile to be built as an ordered representation of spatial proximity among vectors of an unlabelled data set. The data provided by the RFID system is used to train a Kohonen map which allows the definition of a region representing the normal behaviour of the observed subjects. Based on the trained Kohonen map, any activity that does not match the defined normal behaviour is identified as an anomaly; the main advantage is that there is no need to pre-define the pattern of an intrusion. Kohonen's maps are also recognised for their ability to automatically classify activities without supervision. Thus, our anomaly detection system

detects any spoofing attack wherein an adversary mimics an authentic tag or uses a robed tag since such an intrusion will presumably deviate from normal usage.

2.2 Signature-based Intrusion Detection in MANETs

Securing *ad hoc* networks presents a challenge since these networks rely on an open medium of communication, are cooperative by nature and hence lack centralised security enforcement points (*e.g.*, routers) from which preventive strategies are launched. While a wide variety of attacks [118] are aimed at disrupting *ad hoc* networks, routing protocols constitute a key target because: (i) devices operate as routers, which facilitates the manipulation of messages and more generally the compromising of the routing, and (ii) no security countermeasure is specified as a part of the published RFCs. In the following, we introduce an Intrusion Detection System (IDS) that monitors the attacks against routing protocols. We exemplify our IDS, focusing our attention on the attacks that threaten the Optimized Link State Routing protocol [39] (abbreviated as Olsr). We begin by detailing each attack (§ 2.2.1), relying on a model that captures the complexity and temporal dependencies between each of the constituting sub-tasks. We attempt to describe the general form of this attack so as to deal with slightly varying attacks. Based on these modelled attacks, we can further define the corresponding attack signatures (2.2.2).

2.2.1 Attacks Threatening the OLSR Protocol

We start with a brief introduction to the Olsr protocol.

OLSR in a nutshell

Olsr is proactive routing protocol designed to ensure that each terminal maintains an up-to-date picture of the network topology. A fundamental aspect is the Multipoint Relay (Mpr): each terminal selects a subset of 1-hop neighbours whose task it is to forward control traffic to the whole network. OLSR attempts to select the minimum number[a] of Mprs that cover all the two-hop neighbours, so as to reduce the number of nodes re-transmitting control messages and thereby the resulting bandwidth usage. In practice, a node establishes the set of 1-hop neighbours that are further advertised[b] in a so-called *hello message*, which is sent periodically. A Topology Control message (Tc) is then broadcast. Within the Tc message, the Mpr announces the set of neighbours that selected it as Mpr. Thanks to these Tc messages, any device can calculate the shortest path, in terms of number of hops, to any destination. This path is materialised by a sequence of Mprs. In addition, later versions of Olsr support the presence of nodes with multiple network interfaces that are declared in a message called a *Multiple Interface Declaration* (MID) message that is regularly broadcast by Mpr. The functions we have laid out here form the core of Olsr though other extensions have been introduced (*e.g.*, the interconnection to other routing domain thanks to Ospf protocol).

a. Redundant Mpr(s) can be selected to increase accessibility.

b. Instead, information coming from the link layer and typically provided by the IEEE 802.11 protocol, can be used to update the list of 1-hop neighbours.

In the following we detail the attacks to the Olsr protocol, leveraging a model [4] that provides the level of expressiveness needed to depict the actions constituting the attacks along with the

related consequences. We further enrich the model with temporal annotations (as visible in Table 2.1). Then we categorise attacks threatening the OLSR protocol according to actions undertaken against the following routing messages [121]:

- *Drop attack*: consists in dropping routing message(s).
- *Active forge attack*: proactively generates deceptive routing message(s).
- *Modify and forward attack*: modifies received routing message(s) before forwarding it.

Communication			
$X \xleftarrow{M_t} Y$	At time t, Y sends a message M received by X	$\xleftarrow{M_t} Y$	At time t, Y send a message M
$X \not\xleftarrow{M_t}$	X does not receive a message M at time t		
Parameters			
$\triangle t$	Time period	NS_X	1-hop neighbour of X
sq	Sequence number	hc	Hop number
MPR_X	MPRs of X	Sel_{MPR_X}	MPR selector for X
Messages			
$Hello$	Hello message	TC	TC message
FM	Forwarded message	CM	Control message

Table 2.1 – Notations

Drop attack

A Drop attack consists in suppressing a control message that should be normally[1] relayed. A Drop attack is carried out by the MPRs, which forward control messages (*i.e.*,Tc, Mid or Hna messages). Let us exemplify a drop attack: at time t, host H sends a control message, which is intended to be forwarded. The message is received by an Mpr I that does not forward it during a time period that is greater than the maximum allowed period: $\triangle t$:

$$\begin{gathered} I \xleftarrow{FM_t} H, \not\xleftarrow{FM_{t'}} I, |t' - t| > \triangle t \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{2.1}$$

There are few variants of drop attacks: a malicious Mpr may either delete all the incoming control messages (black hole) or a portion of them (grey hole) depending on e.g., the source, the destination or the message type. Attack is detectable by comparing the rate of packets received with that re-transmitted, considering each message type individually. Rather than deleting messages, an alternative behaviour consists in modifying the routing messages prior to forwarding it.

1. Deletion shall exclude the withdraw of some packets, which are empty, expired, duplicated or badly formatted.

Active forge attack

An active forge attack proactively generates misleading routing messages. The best-known example of this kind of attack is probably the Denial Of Service attack (DOS), where a massive amount of control messages are forged to saturate the communication medium (see Expression (2.2)). The attack is usually conducted in a distributed manner with the participation of several devices. The scope of a DOS is either local (*i.e.*, targeting 1-hop neighbours) or global in which case messages are broadcast over multiple hops. While a local attack cannot be prevented, it is recommended to delay and limit the forward of control messages to circumvent a global attack.

$$\begin{gathered} \xleftarrow{CM_t} I, \xleftarrow{CM'_t} I, |t' - t| < \bigtriangledown t \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{2.2}$$

A DOS attack is highly visible and is hence typically conducted along with a masquerade in which the attacks switch their identity. The insertion of falsified messages also falls under the category of active forge attacks. Typically the falsification concerns the list of adjacent links that are advertised in hello messages. Similarly, the information about network interfaces in the MID and HNA messages may be modified. In the first case (Expression (2.3)), I forges a *hello* message, which declares a list of 1-hop and symmetric[2] neighbors NS'_I that differs from the real set NS_I.

$$\begin{gathered} S \xleftarrow{hello(NS'_I)_t} I, NS'_I \neq NS_I \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{2.3}$$

In particular, the basis of an attack aiming at falsifying the state of the links lies in:

- Publishing the presence of a symmetric 1-hop node that does not actually exist (Expression (2.4)). This allows the intruder to be selected as MPR. Indeed, if I advertises a non-existing node N ($N \notin \mathcal{N}$, with $\mathcal{N}$ defining the set of nodes composing the OLSR network[3]), I ensures that no other (well-behaving) MPR claims being a 1-hop symmetric neighbour of N. I is hence selected as an MPR. In addition, the connectivity of I is also artificially increased ($Card(NS'_I \backslash (NS'_I \cap \mathcal{N})) > 0$).

$$\begin{gathered} \xleftarrow{hello(NS_S)_t} S, S \xleftarrow{hello(NS'_I)_{t'}} I, |t' - t| < \triangle t, \\ \exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I \\ \Downarrow \\ I \in \mathcal{I}, \\ \exists I' \in \mathcal{I} \cap NS_S \ni: I' \in MPR_S, \\ Card(NS'_I \backslash (NS'_I \cap \mathcal{N})) > 0. \end{gathered} \tag{2.4}$$

- Publishing an existing node s is a 1-hop neighbour, when it is not the case. This claim disrupts the routing and artificially increases the connectivity of I i.e., $Card((NS'_I \backslash NS_I) \cap$

2. A symmetric 1-hop neighbor, hereafter simply referenced as neighbor, corresponds to an adjacent node with which communication is bidirectional.

3. According to the OLSR RFC [39], messages can be flooded into the entire network (with a maximum network diameter defined by the message Time To Live field, or flooding can be limited to nodes within a diameter (defined in terms of number of hops) from the originator of the message. For the sake of clarity, let be $\mathcal{N}$ represent the network in both case.

$\mathcal{N}) > 0$.

$$\begin{gathered} \xleftarrow{hello(NS_S)_t} S, S \xleftarrow{hello(NS'_I)_{t'}} I, |t'-t| < \triangle t, \\ \exists X \in NS'_I \cap \mathcal{N} \ni: X \notin NS_I \\ \Downarrow \\ I \in \mathcal{I}, \\ Card((NS'_I \backslash NS_I) \cap \mathcal{N}) > 0, \\ \nexists A \in \mathcal{N} \backslash \mathcal{I} \ni: A \in NS_S \wedge X \in NS_A \\ \Downarrow \\ \exists I' \in \mathcal{I} \ni: I' \in MPR_S. \end{gathered} \tag{2.5}$$

If no other (well-behaving) MPR covers S ($\nexists A \in \mathcal{N} \backslash \mathcal{I} \ni: A \in NS_S \wedge X \in NS_A$), then at least one misbehaving node is selected as a MPR of S ($\exists I' \in \mathcal{I} \ni: I' \in MPR_S$). Such insertion typically characterises an attempt to create a blackhole: I introduces a novel path toward M that whenever selected provisions the blackhole.

- omitting a symmetric 1-hop neighbour P as a means of isolating the latter. As a matter of fact, omitting to advertise the presence of P ($\exists P \in NS_I \ni: P \notin NS'_I$) artificially decreases the connectivity of both P and I ($NS_I \nsubseteq NS'_I$).

$$\begin{gathered} \xleftarrow{hello(NS_S)_t} S, S \xleftarrow{hello(NS'_I)_{t'}} I, |t'-t| < \triangle t, \\ \exists P \in NS_I \ni: P \notin NS'_I \\ \Downarrow \\ I \in \mathcal{I}, \\ \exists I' \in \mathcal{I} \cap NS_S, NS_I \nsubseteq NS'_I. \end{gathered} \tag{2.6}$$

The falsification of the adjacency links may pervert the overall local topology, which is perceived by a node and may influence MPR selection. The attack is potentially coupled with a modification of the willingness field [3]. I may prevent (resp. ensure) its selection as MPR by setting willingness field to the value *will_never* (resp. *will_always*). The MPR selection is altered either by falsifying the topological information or the willingness attribute. Overall, the aforementioned forge attack may contaminate the OLSR network ; interconnected routing domains are impacted[4] if a compromised gateway forge falsified routing messages. In such a case the gateway publishes some nodes or some networks that do not exist or exist but are unreachable. In addition, the gateway may also omit to advertise some existing nodes or networks. This attack is quite similar to a link spoofing attack, which is why we will not detail it here. Another way of carrying an attack lies in altering the control messages that are disseminated.

Modify and forward attack

This attack consists in capturing a victim's control message, modifying and then relaying it. Any packet field may be modified, including *e.g.*, the MPR set in TC message, the interface configuration in MID message and routing information injected in HNA message. The attack also involves replaying the packet several times or forwarding it later (possibly to a different location) so that routing tables are updated based on outdated information. Interestingly,

4. Symmetrically, false routes can be imported into the OLSR domain.

attacks may be carried out by multiple nodes. Let us illustrate this case by considering a pair of intruders (Expression 2.7).

$$\begin{gathered} I_1 \xleftarrow{CM(S)_t} S, I_2 \xleftarrow[enc]{CM(S)} I_1, \xleftarrow{CM(S)_{t'}} I_2, \\ \bigtriangledown t \leq |t' - t| < \bigtriangleup t \\ \Downarrow \\ X \in NS_Y \end{gathered} \tag{2.7}$$

One intruder, I_1, records control message from one region while another intruder, I_2, modifies and forwards the message to another region. To do this, I_1 may transmit the message to I_2 using a different interface than that of the ad hoc network. Alternatively, I_1 may create an encrypted tunnel through which encrypted control messages pass. At the other end of the tunnel, I_2 replays incoming messages. The attack creates a *wormhole* whose length corresponds to the distance separating the two intruders. In order to remain invisible, I_2 may masquerade; the source of the relayed control message can be either I_1 – thus I_2 remains invisible – or another node, say S – thus both I_1 and I_2 remain invisible. In order to circumvent the attack, nodes should check if the data transfer is done in time.

Another possible attack is to hijack sequence numbers so that the destination deletes the message that would otherwise have been used. In practice, an intruder I increases[5] (resp. decreases) the value taken by the sequence number so that the destination assumes that I advertises most recent (resp. obsolete) routes and ignores the next (resp. current) control message. Note that if the packet conveying the sequence number is forged then we are dealing with an active forge.

The attacks we have just mentioned can be composed and conducted independently or cooperatively, possibly coupled with a masquerade. Thus devising an intrusion detection system capable of tracking these attacks along with their perpetrators is not a trivial task because the slightest deviation from one attack may make it undetectable.

2.2.2 Attack Detection

The attacks targeting the OLSR protocol follow some well-defined patterns that can be modelled with attack signatures. In the following, we propose an intrusion detection system that aims at monitoring the operation of the OLSR protocol, looking for series of events that match the codified signatures so as to counter security threat. Our IDS is non-invasive: it collects audit data (*i.e.*, logs) generated by the Olsr protocol. Traces correspond to a sequence of chronological events that characterise the evolution of all the internal activities of the protocol (*e.g.*, packet reception or Mpr selection). Additional security reports and logs on the system state are easily integrated and correlated. In addition, our IDS does not require any amendment to the OLSR protocol, which generates separate individual log files. The first step of the online intrusion detection process lies in collecting and pre-processing logs to extract relevant events and avoid matching a attack signature against a large dataset. Events are then classified into the following four categories that reflect the degree of evolution – and hence the seriousness – of the attack:

- *Initial event group*: contains the events that lead to the initiation of an extensive investigation.

5. A sequence number reaching its maximum is reset.

- *Suspicious-evidence-group*: encompasses the events leading to the qualification of a node as possibly malicious.
- *Confirmed-evidence-group*: comprises the event confirming the attack occurrence.
- *Cancel-evidence-group*: includes events that eliminate all suspicion.

Based on a compact set of categorised events, our signature-based IDS supports a fast pattern matching that ensures that incidents are promptly addressed. Additionally, the evolution of a long term attack can be easily tracked. Once filtered and analysed, logs are matched against predefined intrusion signatures. A signature can be viewed as an ordered sequence of events that characterises a suspicious activity. The procedure can be expensive in terms of memory and bandwidth usage as it may require the collection of evidence from other nodes in order to correlate them. Thus, an investigation needs to be initiated if a sufficient degree of suspicion exists. Before delving into the operation of the above evidence groups, let us first exemplify the proposed intrusion detection system with the link spoofing attack we purposely developed.

Signature of a Link Spoofing Attack

Central to the notion of misuse detection is the definition of the attack signature. We are interested in establishing the signature of an attack that tampers adjacent links. A first step towards that goal lies in describing the specific[6] attack relying on the model introduced in [4], which we enrich with temporal annotations so as to support the expressiveness necessary to establish the relationship between actions constituting the attack and the resulting consequences. In order to perform a link spoofing attack, an intruder, denoted I, forges a hello message, which declares the list of 1-hop neighbours whose link is bidirectional, denoted NS'_I and which differs from the correct one, denoted NS_I.

$$\begin{array}{c} S \xleftarrow{hello(NS'_I)_t} I, NS'_I \neq NS_I \\ \Downarrow \\ I \in \mathcal{I} \end{array} \tag{2.8}$$

In order to ascertain the occurrence of such an attack, one should determine whether NS'_I differs from NS_I. There are three variants of falsification (see §2.2.1):

1. Declaring a non-existent node as a symmetric 1-hop neighbour of I (or of another node) so that I (or another node) is selected as MPR.
2. Claiming that an existing node as a 1-hop symmetric neighbour even though the node is distant or is an asymmetric 1-hop neighbour.
3. Omitting a 1-hop symmetric neighbour artificially decreases the connectivity of the omitted node.

The detection of the link spoofing attack described above necessarily requires a continuous investigation that aims at confronting each node's vision (which includes the set of 1-hop and 2-hops neighbours). Unfortunately this involves a continuous confrontation and hence is not viable. We therefore take a different approach, noting that inflecting the MPR selection is one

6. Interested reader may refer to [8] for a detailed modelling of all the attacks threatening OLSR and a description of the related signatures.

attack goal and thereby an attack indicator. Our approach thus consists of looking for changes (or an absence of changes) in the MPR selection and in the MPR coverage. As a complement, a MPR behaving inappropriately (*e.g.*, suppressing, falsifying, failing to adequately relay control messages or defining itself as a MPR without having been selected as such) is also an interesting aspect. Events (extracted from the logs) that reveal such (possibly inappropriate) behaviour are tagged as evidence announcing an attack occurrence and are categorised as an "initial event group" (as defined in §2.2.2). In such a case, an in-depth investigation is triggered to collect information concerning the 1-hop and 2-hops neighbourhood.

Collaborative Investigation An in depth investigation consists in verifying that the MPR coverage is adequately advertised. For this purpose, 2-hops neighbours are questioned with the aim of determining if the suspicious MPR is indeed a symmetric 1-hop neighbour and is susceptible of being a MPR. The corresponding request is sent out, avoiding the suspect as much as possible. Based on the returned answers, the attack diagnosis takes place using the signature described above. If all the interrogated nodes provide information that conforms to that of the suspected MPR, the behaviour of the MPR is defined as appropriate. If instead one or more nodes provide information that differs from the MPR information, this may indicate that an attack is taking place. Two special cases in particular may pose problems:

- Some or all of the responses are missing due to *e.g.*, some packet collisions or some disconnections of the queried nodes. In such a situation, the diagnosis is biased as it is only based on a portion of the evidence.
- Answers are contradictory. For instance, a malicious node may non-legibly accuse a well-behaving MPR or may certify the good behaviour of a malicious MPR. A legitimate node may also have a different vision due to *e.g.*, the node mobility or to slightly inconsistent routing table.

In order to prevent misbehaving nodes from foiling the intrusion detection, we propose to evaluate the trustworthiness of the node(s) that provide second-hand observations (see § 2.2.3). The objective is to favour the observations provided by trustworthy nodes while being detrimental to misbehaving nodes. In addition we leverage a statistical parameter, namely the confidence interval, to control the evidence-gathering and determine to what extent additional evidence is needed to make an informed judgement (§ 2.2.4).

2.2.3 Trustworthiness Evaluation

In the presence of malicious nodes, supporting a reliable intrusion detection strategy is challenging because attackers will try to conceal the attack by providing false observations. While essential, trust establishment in a distributed and resource-constraint MANET is much more difficult than in traditional wired networks as there is no certification authorities nor trusted third parties; as opposed to traditional centralised approaches, trust management in MANETs should handle uncertainty and incompleteness of trust evidences.

In the years 2010, a great majority of existing work on distributed intrusion detection assumes that any participating node is trustworthy and faithfully reports intrusion attempts [63]. Alternatively, Duma et al. [55] introduce a trust-aware engine that correlates intrusion alerts. Nonetheless, the proposed trust model does address highly distributed intrusion detection. Meanwhile, much research has been conducted on modelling and quantifying trust [87]. Trust and reputation are usually utilised to prevent packets from being routed through misbehaving

nodes [151, 175]. In this work, we develop a robust trust management model that specifically copes with distributed intrusion detection. Our trust system evaluates node trustworthiness based on personal experience and on the recommendations formulated by others. We propose a distributed and entropy-based trust system in which trust relationships are primarily built with neighbours, based on local observations and on mutually exchanged evidence. Trust is evaluated depending on the similarities between received evidence and local evidence. Each time a node provides a similar (resp. dissimilar) evidence of intrusion, its trustworthiness increases (resp. decreases). Recommendations from neighbours are also necessary to derive indirect trust values. A fundamental challenge is then to define a suitable means by which to represent trust and synthesise the set of opinions that are provided by others into a unique aggregated value. To this end, we first define the notion of trust and introduce some axioms that describe the basic rules for establishing trust. Then, based on these axioms, we develop techniques to calculate trust values.

Trust Definition

A trust relationship established between node A concerning node I, represents the extent to which A thinks that I behaves adequately. Trust is established before a misbehaving action takes place and as such can be viewed as a belief, i.e., level of likelihood with which a node is expected to perform a particular action. Trust is characterised by the following properties:

- Trust is subjective. For instance, the trust that two nodes grant to a third node may differ.
- Trust has a dynamic nature and usually varies over time. The trust value is a discrete real number.
- Trust is an action – and context– dependent function: an entity may be trustworthy for performing one task (*e.g.*, forwarding packets) but not for another one (*e.g.*, detecting attacks).
- Trust is asymmetric and not transitive. Trust is not necessarily mutual and the fact that A trusts B and B trusts C does not imply that A trusts C,

Building trust requires a quantitative analysis of the nodes behaviours based on the history of their previous activities and interactions. Thus, trust can be assessed by a measure of uncertainty, and as such trust values can be estimated based on entropy. In the following we introduce the rules (axioms) to follow for calculating trust values based on observations and also through a third party (concatenation propagation) and through recommendations from multiple nodes (multipath propagation).

Trust Establishment

The following five axioms are used to establish trust relationships accordingly, leverage previous interactions and consider recommendations:

- **Axiom 1 - Measuring trust based on entropy:** An activity that is beneficial to others (*e.g.*, packet relaying) increases the confidence in the node that performs it. Conversely, a malicious activity decreases this confidence. A piece of evidence that concerns a node I and is established by a node A, is denoted $e_j^{A,I}$. Such evidence contains all the observations necessary to evaluate the trust and calculate a value that reflects the confidence associated

with the evidence. The confidence associated with the evidence is not absolute but the opinion of the specific node instead. In order to increase the readability of the manuscript, the trust value is referred as an evidence value. The evidence (a.k.a confidence value) takes a positive value (resp. a negative value) if the related behaviour is beneficial (resp. malicious). The trust is a real number in [-1,1]. When the trust value is -1 the node is not trusted and if trust value is +1 the node is trusted. We define the entropy-based trust value as follows:

$$T^{A,I}(e_j^{A,I}) = -e_j^{A,I}\ log_2(e_j^{A,I}) - (1 - e_j^{A,I})\ log_2(1 - e_j^{A,I}) \tag{2.9}$$

- **Axiom 2 - Danger:** the degree of danger should be taken into account when assigning a trust value. To do so, a factor α_j weights the evidence $e_j^{A,I}$.

- **Axiom 3 - Freshness:** Newer activities should be prioritised over older ones. In practice, an omission factor β privileges fresh evidences rather than older ones. A node A calculates the trust of a node I based on n evidence denoted $e_1^{A,I}, \cdots, e_i^{A,I}, \cdots, e_n^{A,I}$, which concerns I and that have been collected during Δt:

$$T_{\Delta t}^{A,I} = \sum_{j=0}^{n} \alpha_j\ e_j^{A,I} + \beta\ T_{\Delta t-1}^{A,I} \tag{2.10}$$

  Note that in absence of evidence, the trust value is periodically updated as follows to ensure node redemption: $T_{\Delta t}^{A,I} = \beta\ T_{\Delta t-1}^{A,I}$

- **Axiom 4: Recommendation of a third party:** First-hand evidence that is obtained by the node itself is preferred to second-hand evidences, which are comparatively more controversial. Nonetheless, when the observations obtained by A are not sufficient, additional (less reliable) evidence provided by other nodes is gleaned (see § 2.2.4). In this case A considers the recommendations of the third parties but the uncertainty increases. Node A establishes the trust value associated with a node I based on the recommendation of a third party S as follows:

$$Ts_{\Delta t}^{A,I} = R_{\Delta t}^{A,S} T_{\Delta t}^{S,I} \tag{2.11}$$

  Thus node A relies on the evidence $T_{\Delta t}^{S,I}$ provided by S. Given that node S may lie, A lowers the resulting trust value using $R_{\Delta t}^{A,S}$.

  Note that the computation of the trust value (Eq. 2.11) reflects the probability that I will perform an attack. Let p_S denote the probability that S makes a correct recommendation and $p_{I/B=0}$ the probability that I will perform an attack if B lies. This probability can be expressed as :

$$p_{ASI} = p_S\ p_{I/S=1} + (1 - p_S)\ p_{I/S=1} \tag{2.12}$$

  Unfortunately A ignores the value of p_S and $p_{I/S=1}$. Leveraging our distributed trust model, A evaluates the probability $p(AS)$ that A observes the behaviour of S and A makes a recommendation on S. In addition, A estimates $p(SI)$ the probability that S observes the behaviour of I and that S makes a correct recommendation on S. Finally, A computes the value of p_{ASI}:

$$p_{ASI} = p(AS)\ p(SI) + (1 - p(AS))\ (1 - p(SI)) \tag{2.13}$$

- **Axiom 5- Concatenating multiple recommendations:** The recommendations of

several nodes may be considered together in order to establish a trust value. Recommendations are concatenated into a trust value that should not be artificially amplified. Thus, when several nodes, denoted $S_1, S_2, \cdots, S_m$, generate recommendations, A computes an aggregated trustworthiness about I as follows:

$$Tm_{\Delta t}^{A,I} = \sum_{i=1}^{m} w_i R_{\Delta t}^{A,S_i} T_{\Delta t}^{S_i,I} \tag{2.14}$$

with $w_i = \frac{1}{\sum_{j=0}^{m} R_{A,S_j}^{\Delta t}}$ serving as an averaging factor. Once established, trust is then used to prevent malicious nodes from interfering with intrusion detection.

Trust Establishment in the presence of a link spoofing attack

In the following, we detail the computation of the trust value considering a link spoofing attack: In order for Node A to establish whether I is carrying a link spoofing attack A interrogates the neighbours of the suspected node I. The interrogated neighbours either do or do not corroborate the information provided by the suspect, in which case the answer includes the fields $r^{S_i,I}$ that either takes the value 1, if the link established by I is correct, or the value -1 otherwise. The answers from each interrogated neighbour S_i (with $i \in [1, m]$) are weighted with the trust level T_{A,S_i} that the investigator A places in the neighbour S_i as well as taking into account a weighting factor w_i:

$$Detect_{\Delta t}^{A,I} = \sum_{i=1}^{m} w_i T_{A,S_i} R_i^{S_i,I} \tag{2.15}$$

with $w_i = \frac{1}{\sum_{j=0}^{m} T_{A,S_j}}$. Ultimately, a fake link attack is detected if $Detect_{\Delta t}^{A,I}$ is close to 1.

2.2.4 Reducing the Cost of Investigations

Intrusion detection is a resource-consuming process that needs to last because an attack against a routing protocol is typically persistent and continuous (*i.e.*, attack repeats itself over time). Thus intrusion detection leads to the collection of a massive amount of logs and evidence. In a dense network, the number of collected evidence becomes colossal. In order to reduce the corresponding bandwidth usage and thereby lighten the intrusion detection process, we propose to randomly and uniformly request the nearby nodes into collaborative investigation (*i.e.*, the 2 hop neighbours of the MPR). As a result, fewer nodes are less frequently contacted compared to the deterministic approach introduced above. We employ confidence interval and confidence level as means for regulating (either reducing or increasing) the evidence-gathering while ensuring a satisfactory level of detection reliability.

Confidence interval and confidence level

The *confidence interval* represents a statistical uncertainty associated with the estimation of a population parameter (*e.g.*, proportion, mean, median) [156]. The confidence interval is associated with *confidence level*, which corresponds to the likelihood that the true parameter belongs to the interval.

In our case, based on a partial set of evidence, denoted $e_1, \cdots, e_n$, we establish an interval in which the mean, computed using the entire set of evidence, has a high probability of belonging to. In practice, based on a confidence level fixed by the investigator, the true mean $\bar{m}$ has a certain probability of being contained in the confidence interval $[\bar{m}_e - \epsilon,\ \bar{m}_e + \epsilon]$ with ϵ

reflecting the allowed margin of error that is defined by the end user. The sampling error ϵ decreases as the sample size increases. According to the central limit theorem introduced by Pierre Simon Laplace [135], the sampling distribution of the mean follows a t-distribution [157] when the sample size is small and a normal distribution when the sample size becomes large [11], regardless of the population form. With a normal distribution, the margin of error ϵ is expressed as a function of the standard deviation σ and the standard density z:

$$\epsilon = z\frac{\sigma}{\sqrt{n}} \tag{2.16}$$

With a t-distribution, the margin of error ϵ is expressed as a function of the standard deviation σ and the standard density z:

$$\epsilon = z\frac{\sigma}{\sqrt{df}} \tag{2.17}$$

with df corresponding to the degree of freedom. Note that there are many potential t distributions, whose forms are determined by their degrees of freedom. Nonetheless, when estimating the mean, $df = n - 1$. The standard deviation is then calculated as follows:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n}(\bar{m} - e_i^{A,S_i})^2}{n-1}} \tag{2.18}$$

The overall estimation of the confidence interval of a population mean proceeds as follows:

- Gather a few pieces evidence (a.k.a samples).
- Compute the sample mean and standard deviations.
- If the sample size is large (resp. small), select the $z_{\alpha/2}$ (resp. $t_{\alpha/2}$) from the normal distribution (resp. the t-distribution),
- Compute the sampling error ϵ and the confidence interval as $[\bar{m}_e - \epsilon, \bar{m}_e + \epsilon]$.

The computation of the confidence level and interval during the diagnostic permits our Ids to: (i) regulate the number of gathered and processed evidences while maintaining the required detection accuracy and (ii) measure to what extent the diagnosis is reliable, especially in the presence of inconsistent evidences.

At best, the confidence level is high and the confidence interval is narrow. If an attack[7] is taking place, evidence will be collected with less but sufficient regularity. In practice, the IDS follows a stepwise process that reduces the volume of evidences to collect. Instead, if the confidence level is low and the confidence interval is wide, additional evidence should be collected to establish a representative diagnostic. Note that collecting more evidences conveniently increases the confidence level but not necessarily narrow the confidence interval: a controversy is established but cannot be prevented.

2.3 Anomaly Detection

We propose an anomaly detection, whose aim is to find any behaviour that does not conform to the expected behaviour. We begin by modelling the normal behaviour of the target and define an anomaly as any observed behaviour that does not conform the the normal/baseline model. A

7. In absence of detected attack, investigation ends.

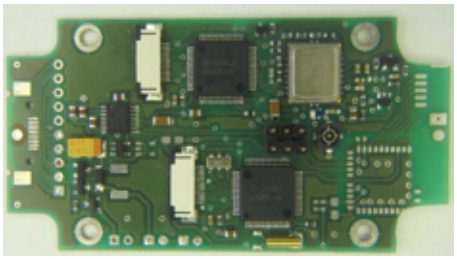
key challenge to searching for an anomaly lies in identifying which data are dissimilar to the rest of the data set, since an individual piece of data can be considered as anomalous with respect to the the larger whole. As an illustration, an anomaly may refer to credit card transaction characterised by a very high amount of expense compared to the usual range of expenditure. A contextual anomaly corresponds to a data instance that is anomalous only in a specific context. A collective anomaly comprises a collection of instances; the individual instances are not by themselves defined as anomalies but their concomitant occurrences, as a collection, is abnormal. A Denial Of Service illustrates such an anomaly: individual events are not anomalies when they occur sporadically over a large time frame.

In order to detect individual, context-dependant and collective anomalies, we investigate the feasibility of applying machine learning techniques and in particular unsupervised learning, considering an application that controls the access of users [138, 126]. To this ends, users are provided active RFID tags which are used to monitor the user's location. Given this specific use case, anomalies are primarily detected based on spatial-temporal features rather than operational ones. Rather than relying on a simple statistical method, we select an advanced neural network architecture permitting the construction of the user's profile automatically *i.e.*, without involving users or experts. Note that such automatic training permits the easy addition of (operational) features without modifying the core implementation. Based on an advanced Kohonen's map, our detection system detects any spoofing/cloning attack wherein an adversary mimics an authentic tag and any usage of a robed tag since these intrusions, by assumption, will deviate from normal user's normal activity.

2.3.1 RFID System supporting access control

We propose an anomaly detection model that deals with an access control application in which users working within a building are equipped with RFID tags which are used to monitor the user's location. The application records the user's profile including the series of Cartesian coordinates of that user so as to detect any intruder in the building.

(a) RFID tag.



(b) RFID reader.

Our access control application [138, 126] consists of three main components:

- An RFID tag (Figure 2.1a)– a silicon microchip attached to an antennae and possibly enriched with additional functionalities, *e.g.*, sensing, storage, encryption – that outfits the users.
- An RFID reader (Figure 2.1b) –a transceiver communicating with tags via radio frequency and typically containing internal storage and processing capabilities (so as to perform tasks on behalf of the tag), which is deployed within the building,
- A back-end database connected to the reader, which collects the information related to the physically tagged persons.

The above components are naturally subject to a broad range of threats [123], due to the networked nature of RFID, their poor physical security as well as their insufficient resilience against physical manipulation.

Security Flaws

A particularly common issue is related to the fact that RFID tags may be permanently or temporally disabled. Possible ways of rendering a tag inoperable involve: (i) destroying the tag, (ii) disabling the tag using the so-called kill command, (iii) covering the tag with an aluminium shield that serves as a Faraday Cage, and (iv) preventing tags from communicating with readers by generating an electromagnetic signal in the same range as the reader so as to cause jamming. In order to safeguard RFID systems against low-tech attacks that (permanently or temporarily) disable tags, additional countermeasures can be used such as increased physical security through guards and cameras.

Another type of attack lies in *cloning* and *spoofing* an RFID tag. *Cloning* consists in replicating the legitimate tag, which is reasonably easy given the wide availability of writable and re-programmable tags. With a *spoofing* attack an adversary impersonates a valid RFID tag to gain privileges. Spoofing requires a certain expertise; the attacker must (i) understand the involved protocols, and (ii) gain access to the secrets (*e.g.*, keys) used during the authentication. Nonetheless if the RFID communication is unauthenticated, attackers may easily counterfeit the identity of a legitimate reader so as to elicit sensitive information or modify data on RFID tags.

The wireless nature of RFID makes eavesdropping one of the most serious threats. By eavesdropping an attacker records communications (tag-to-reader and reader-to-tag) between legitimate RFID tags and readers. The feasibility of this attack depends in particular on the distance of the attacker from the legitimate RFID devices. For instance, the clandestine scanning of the tag may take place wherever the read range permits it to do so. Such scanning remains undetected (recall that a tag responds to the reader interrogation without alerting its owner). In addition, once a reader powers a tag another reader may monitor the resulting tag emission without putting a signal out itself, *i.e.*, it eavesdrops within the detection range. A misbehaving reader that harvests information from a well-behaving tag is the starting point of privacy concerns, especially when the tag's serial numbers are combined with personal data. Considering the fact that a significant portion of RFID tags employ user-writeable memory, an adversary can exploit this to modify or delete valuable information.

Generally speaking, the level of sophistication of RFID tags and readers depends of their relative financial cost, which may impair their ability to provide advanced encryption and authentication functions. This requires providing intrusion detection. In order to meet this requirement we propose an anomaly detection system that identifies three types of attacks: stolen, cloned and spoofed tags.

2.3.2 Anomaly Detector

Rather than simply relying on the simple statistical method, we select the Kohonen's self-organising maps [92] as an advanced neural network architecture that can build the users' profiles as an ordered representation of spatial proximity among vectors of an unlabelled data set. The justification for this choice is twofold: (i) Kohonen's maps automatically categorise the inputs provided during the training phase without supervision, and (ii) they allow easy enrichment of the user profile, *i.e.*, without necessitating substantial implementation changes. Consequently, based on an advanced Kohonen's map, our detection system identifies a spoofing

attack wherein an adversary mimics an authentic tag and any usage of a robed tag because these intrusions, by assumption, deviate from the normal usage of the users.

The primary challenge in anomaly detection lies in defining normal versus abnormal behaviour. An advantage of self-organising maps is that they can learn to discriminate between normal and abnormal based on examples (*i.e.*, training samples) and therefore requiring no explicit definition of these behaviours by the user. Our anomaly detection system is based on the Kohonen map [92], a neural network that distinguishes itself by its unsupervised learning. Another convenient aspect is related to the fact that this map reduces the dimensions of the input data from a (potentially) high dimension into 2- or 3-dimensional space (here 2-dimensional) therefore allowing an easy visualisation and instinctive interpretation of the results.

The visible part of a Kohonen's map consists of neurons initially arranged in a grid. Each neuron is associated with a weight vector that visually corresponds to the neuron position in the grid. The training consists in moving the neurons; this movement is based on attraction strength determined by neighbourhood function. This neighbourhood function conveniently preserves the topological properties of the input space and serves as a regularisation factor that smooths the functional mapping. The low dimensionality of Kohonen Maps renders them useful for visualisation. The neighbourhood function serves ordering the map by pulling units close together in the map space toward each other. The resulting trained map reflects a mapping from a higher-dimensional input space to a lower-dimensional map space. The self-organised map serves partitioning the input space into convex regions of activity that are characterised with the following property. Every point in the space is closer to the centroid of its region than to the centroid of any other region. In the map, the centroids of the regions are defined by the weight vectors.

As detailed in the following, anomaly detection identifies activities that vary from an established pattern, following three main steps: (i) pre-processing the gathered data, (ii) creating a knowledge model made up of the (previously) monitored activities, and (iii) subsequently categorising the various activities relying on Kohonen maps.

Raw Data Pre-processing

The output of the RFID system corresponds to the trajectory of any subject sampled at discrete time intervals $t_1, \cdots, t_k, \cdots, t_m$ with m defining the trajectory length. Any observation is expressed as a set of m n-dimensional real vectors $x = (x(t_1), \cdots, x(t_k), \cdots, x(t_m))$. A trajectory is hence composed of spatial-temporal records, each record being primarily composed of a geographical location in a 3D plan and a temporal attribute, *i.e.*, a timestamp. In addition to the above, extra pieces of information may be added or inferred from the spatio-temporal records. For instance, they relate to the maximum speed, (estimated) attraction point, direction, movement pattern (*e.g.*, loop, u-turn) and the average or standard deviation of the aforementioned parameters. Data provided by the RFID system are further filtered and normalised so as to fall in a specific $[0, 1]$ range. In practice, filtered samples are expressed as a vectors set in which each vector $x = (x(t_1), \cdots, x(t_k), \cdots, x(t_m)) \in \mathbb{R}^m$ are collected at $t_1, \cdots, t_k, \cdots, \cdots, t_m$. Each activity $x(t_k)$, is defined as an n-dimensional vector $x^T(t_k) = (x_1(t_k), \cdots, x_i(t_k), \cdots, x_n(t_k))$, which once normalised is denoted $x'^T(t_k) = (x'_1(t_k), \cdots, x'_i(t_k), \cdots, x'_n(t_k))$ with $x'_i = \frac{x_i(t_k}{arg\ max_{j\in[1,n]}(x_{ij}(t_k))}$. Relying on these filtering and normalisation processes, work-less samples are removed and each filtered sample is of equal footing and can hence be exploited during the training phase in order to create a Kohonen map.

Training phase

A training phase is needed in order to generate a map. The resulting Kohonen map $w_1, \cdots, w_s$ of size s corresponds to a topological 2-dimensional array of neurons originally initialised with random values. This map is intended to categorise the normalised samples $x'(t_k)$ (with $1 \leq k \leq m$) provided as input. Each input vector $x'(t_k)$ is therefore compared with each neuron forming the Kohonen map and the distance between the input vector and neuron is computed and the closest neuron is selected as the winner. The topological structure of the Kohonen map is then updated: neurons that are topologically close to the winner move towards it. Consequently the resulting Kohonen map reflects a categorisation (clustering) of the samples. More specifically, considering a measure whose norm is noted $||$, the distance between an input vector $x'(t_k)$ and the synaptic vector of each neuron $w_i(t_k)$ of the map is computed and the winner $g(x'(t_k))$ selected according to the following law:

$$g(x'(t_k)) = argmin_{i \in [1,s]} ||x'(t_k), w_i(t_k)||. \tag{2.19}$$

Afterwards the neurons that are topologically close to the identified neuron move in the direction of the winner. To achieve this the neuron w_i is updated as follows:

$$w_i(t_{k+1} = w_i(t_k) + \pi_{i,g(x'(t_k)}(t_k)\eta(t_k)x'(t_k) - w_i(t_k) \tag{2.20}$$

with $(i, j, k) \in [1, s]^2 \times [1, m]$, $\eta(t_k)$ defining an adaptation factor that controls the degree of change imposed to the neuron's vector, and $\pi_{i,g(x'(t_k)}(t_k)$ a neighbouring function centred around the winner $g(x'(t_k)$.

The basic idea is that the adaptation factor $\eta(t_k)$ decreases monotonically as the learning phase progresses so as to guarantee a convergence of the weighted neuron's vector towards a stable state [98]. To this end $\eta(t_k) = \eta_0 \ exp(t_k/t_m)$. Similarly, the neighbouring function $\pi_{i,g(x'(t)}$ decreases as t evolves until the winning neuron is the only neuron that has its weight significantly updated. For this purpose $\pi_{i,g(x'(t)}$ is defined as a symmetric function following a Gaussian form with a standard deviation σ decaying exponentially with time:

$$\pi_{i,g(x'(t_k)}(t_k) = exp\left(\frac{||(x'(t_k), w_i)||}{2\sigma^2(t_k)}\right) \tag{2.21}$$

and,

$$\sigma(t_k) = \sigma_0 \ exp\left(\frac{-t_k \ log(\sigma_0)}{t_m}\right) \tag{2.22}$$

Kohonen algorithm is applicable to large data set given that: (i) the computational complexity scales linearly with the number of samples m, and (ii) limited memory is necessary to record the set of training vectors $x'(t_1), \cdots, x'(t_k), \cdots, x'(t_m)$ and the Kohonen map $w_1, \cdots, w_s$). As a 2D-grid a Kohonen map is of great help in visualising and inspecting the user behaviour recalling that the structure of Kohonen map reflects the structure of the original training samples. By employing the trained Kohonen map, which reflects the normal activity of a subject, any deviation from that normal activity can be easily detected and identified as an anomaly.

Decision Making

If the distance between the observed and normal behaviour is greater than a given threshold, then the observed behaviour is intuitively defined as anomalous. Given our use case - RFID-

enabled control access system attempting to analyse the user location - we distinguish two sources of potential anomalies: the user's position and its trajectory. A position is said to be anomalous if it does not belong to any of the classification defined during the training, *i.e.*, if it does not pertain to any of the clusters centred on the wining neurons defined as part of the training phase. By extension we define a trajectory as anomalous if a large percentage of the user's positions are anomalous, *i.e.*, if the pre-processed observations do not pertain to any of the clusters centred around the winning neurons $g(x'(t_k))$ and delimited by the radius defined as the maximum distance separating the winning node $g(x'(t_k))$ from its neighbouring neurons (*i.e.*, the neurons that belong to $Dg(x'(t_k))$). By extension, a trajectory $o'(t), \cdots, o'(t+p)$ is anomalous if the ratio of anomalous positions exceeds a given threshold defined by $\beta(r)$.

The computational complexity related to detecting a position and then its trajectory scales linearly to the number of winning vectors $g(x'(t_k))$ (bounded by m). In addition to the memory allocated to the training phase, little additional memory (essentially just the index i of the winning neurons and their established radii) is used during the anomaly detection.

Identifying trajectories that exhibit irregular or even suspicious traits is crucial in many applications including access control and intrusion detection. In such scenarios, unsupervised learning is the most advantageous model when learning from many regular data instances, as the algorithm thoroughly approximates the underlying distribution and produces a concise model of normality. As a consequence, unsupervised learning method are often used to detect anomalies.

2.4 Performance Evaluation

In the following section we evaluate performances associated with the intrusion detection system (§ 2.4.1) and the trust system (§ 2.4.2).

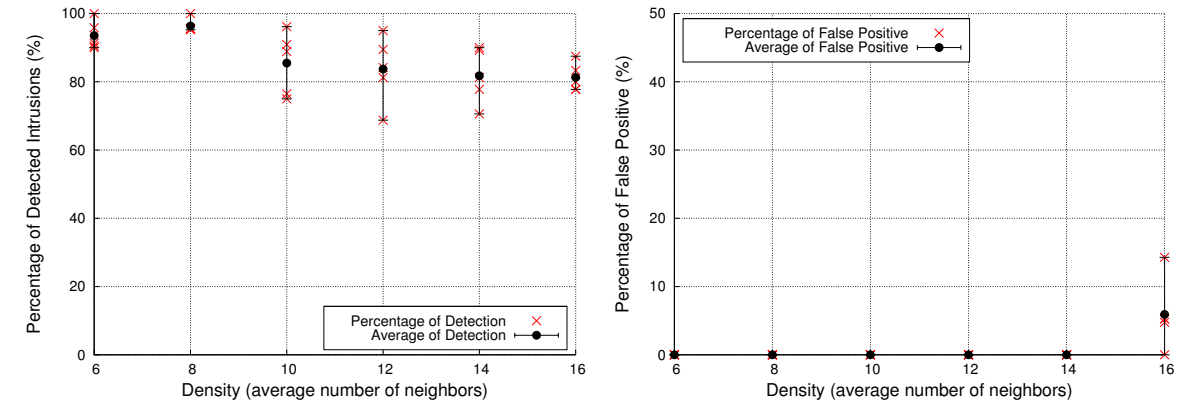
2.4.1 Signature-based Intrusion Detection

In order to evaluate the performance of our intrusion detection system (§2.2) we simulate a mobile *ad hoc* network using the Ns3 network simulator[8] [133] and we virtualise each device using Lxc[9] [23]. The MANET area is defined by a squared area of $S = 310 \times 310\ m^2$ and is comprised of $N = 30$ devices split into 25 well-behaving devices and 5 intruders, the latter of whom repeatedly launch the implemented link spoofing attack[10]. Devices move randomly, following a randomly chosen direction, at a given speed that is the same for all the devices. When a device hits the network boundaries it rebounds following a reflexive angle. Nodes use the Olsr protocol, communicating via Ieee 802.11a with transmissions that have a range of $90m$. Data traffic is further simulated using the *V4PingHelper* application of NS3; the nodes exchanges 56 bytes Icmp echo requests to one another and wait for $1s$ before sending it again.

During our experiments we evaluate the performance of our IDS in terms of:

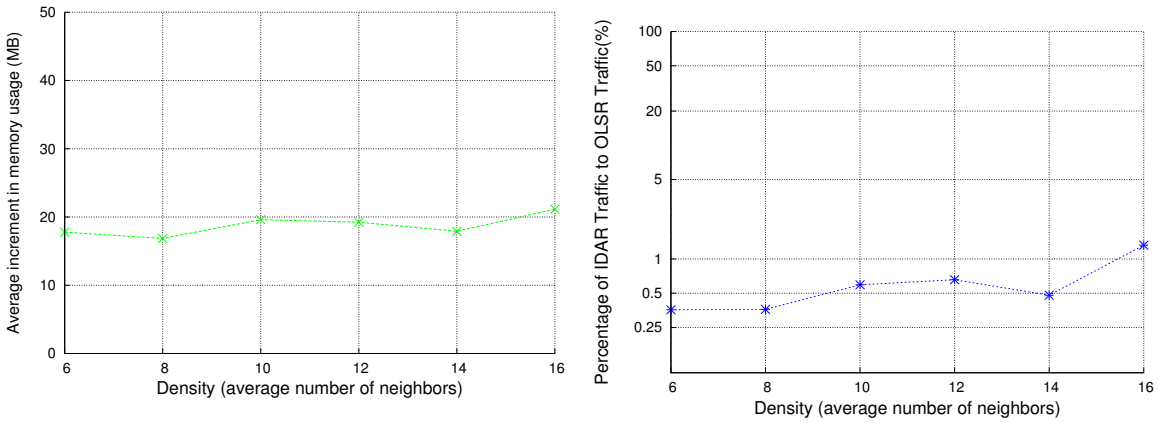
- The number of intrusions that are successfully detected.
- The number of false positives that occur when a legitimate node is wrongly designated as an intruder.

8. http://www.nsnam.org
9. http://lxc.sourceforge.net
10. In our experiments, the number of successful intrusions varies between 15 and 33, according the specificity of the simulated network.



(a) Detection accuracy.

(b) False positive rate.



(c) Memory usage.

(d) Bandwidth usage.

Figure 2.2 – Performances of our intrusion detection system depending on the Network Density.

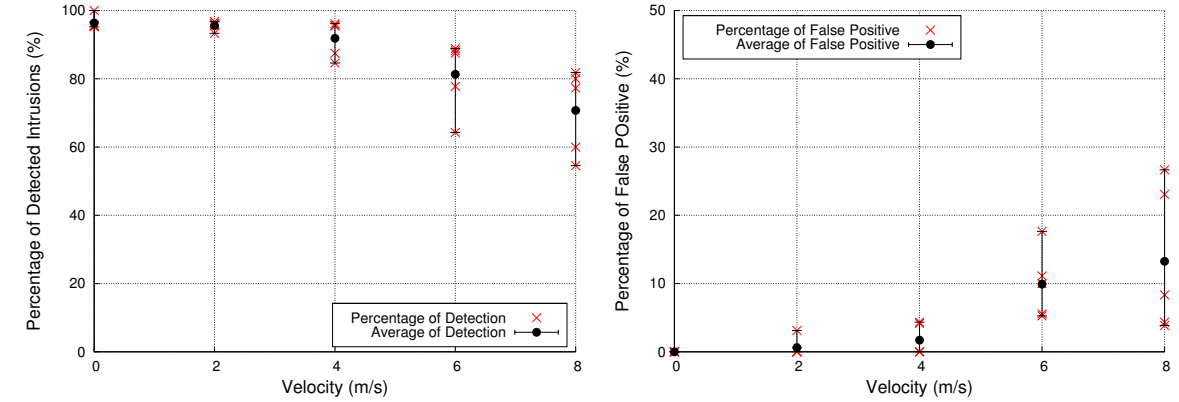
- Detection overhead, referring to the memory usage and network traffic that are generated by our IDS.

Using these performance criteria, we consider the impacts of the network density (Figure 2.2) and the device mobility (Figure 2.3). The network density corresponds to the average number of neighbours, which is defined in [75] as $\frac{N \times \pi \times T_x^2}{S}$.

Impact of the density

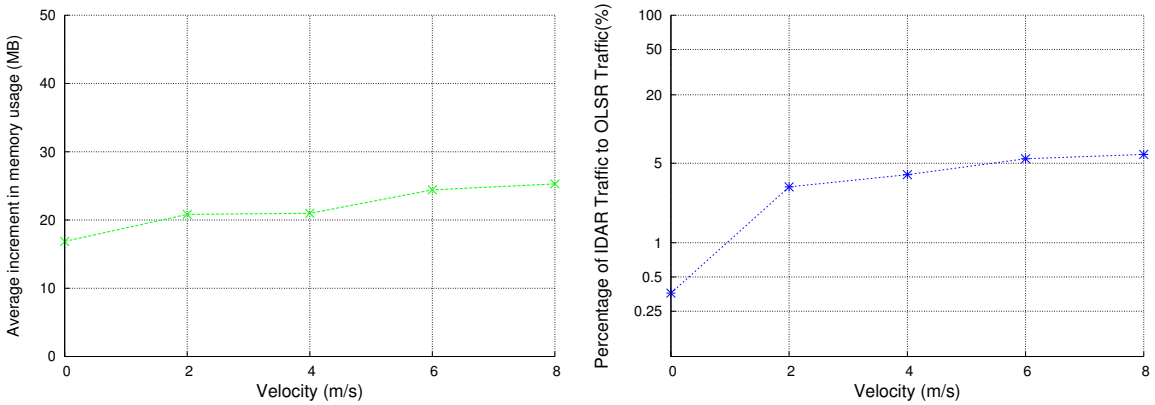
At first glance, the rate of intrusions that are detected is conveniently high (Figure 2.2-a) and the false positive rate remains low (Figure 2.2-b). Specifically, the detection rate increases slightly from 93.5% to a maximum of 96.3% when the density varies from 6 to 8 neighbours because the attack is seen by a larger number of neighbours. Then a slow decrease is observed when density is above 8. The decline is caused by a higher number of collisions that partially prevent the collection of evidence and slows down intrusion detection. Still the detection rate always remains greater than 80%. As visible in Figure 2.2-b, the percentage of false positives is on average always under 5.9%. Further analysis reveal that many false positives occur due to the lack of synchronisation among nodes: some links are valid (i.e., existing) for some nodes and expired (*i.e.*, absent) for others. To address this problem, the period during which links are considered valid should be slightly lengthened, as in [164]. While memory usage (Figure 2.2-c) gently fluctuates between $17.6MB$ and $22.2MB$, the traffic generated by our intrusion

detection system (Figure 2.2-d) is almost negligible. Overall, there is clearly a good trade-off between detection accuracy and resource consumption (*i.e.*, computing and bandwidth usage) associated with the distributed detection. When the density is high, our intrusion detection system is confronted with a typical limitation of a constrained network: the traffic volume – most of which stems from the routing protocol and the application – is too high and cannot be supported by the network. In this situation, MANET can no longer fulfil its mission and intrusions are hardly discerned.



(a) Detection accuracy.

(b) False positive rate.



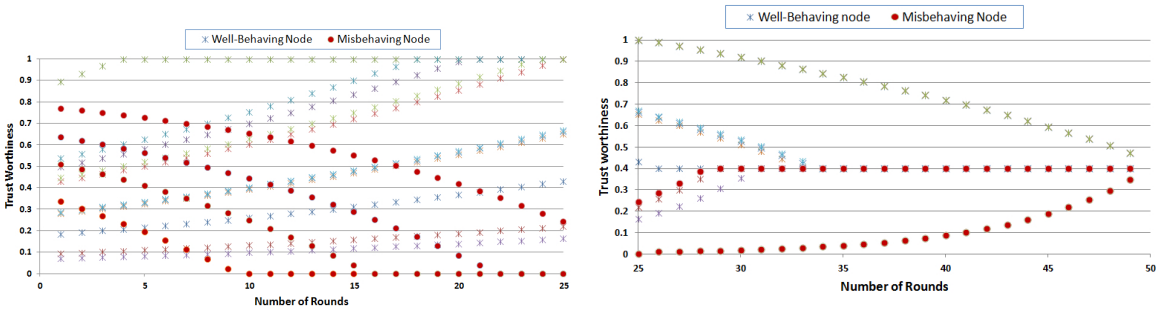
(c) Memory usage.

(d) Overload.

Figure 2.3 – Performance associated with our intrusion detection System, considering a varying Mobility given a network density of 8 neighbours.

Impact of mobility

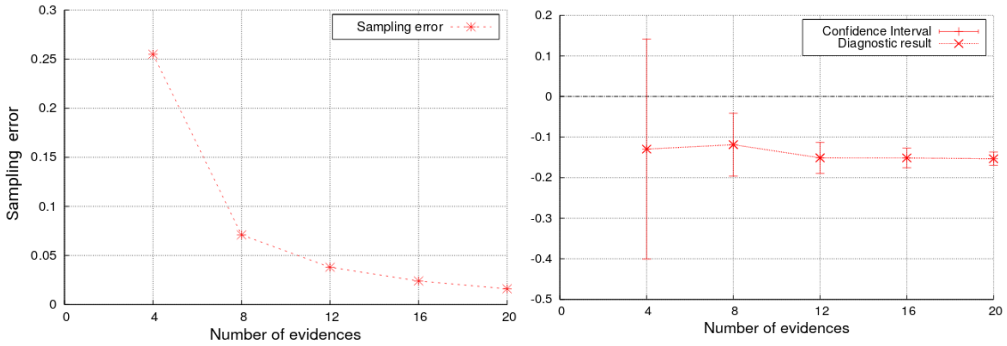
From the standpoint of intrusion detection, mobility is the biggest challenge. When node mobility increases, the intrusion detection rate decreases (Figure 2.3-a) and false positive rate (Figure (2.3-b) falls due to the constantly changing topology. However a high detection rate remains (e.g., about 70.7% with a moving speed of $8m/s$, i.e., around 28.7 kilometers/hour equivalent to 17.89 miles/hour). The memory usage (Figure 2.3-c) rises steadily from $17.6MB$ to $26.5MB$, which is a reasonable figure even for resource-limited devices. Traffic remains very low compared to the Olsr. Overall, our evaluation shows that the intrusion detection system is characterised by a high detection rate and a low false positive rate even under harsh conditions. While the traffic overhead is negligible compared to OLSR, the fairly small memory usage means that the intrusion detection can be undertaken by resource-limited devices.



(a) Trustworthiness.

(b) Impact of the forgetting factor on trustworthiness.

Figure 2.4 – Trust values computed by the attacked node.



(a) Sampling Error.

(b) Intrusion Assessment.

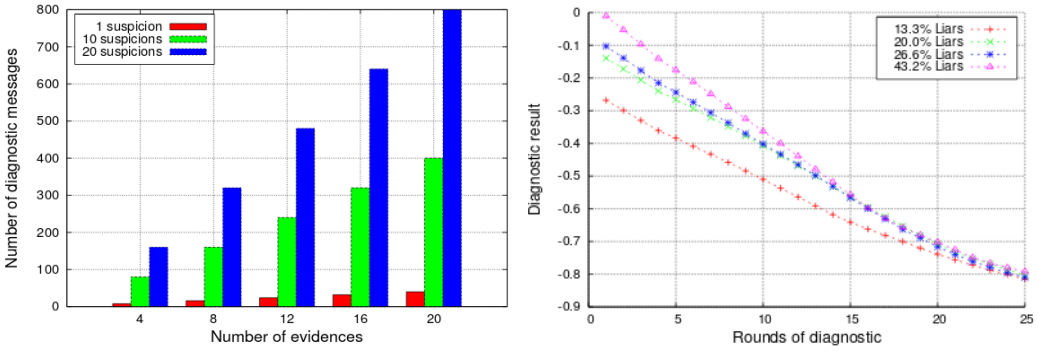
Figure 2.5 – Confidence associated with attack assesment.

2.4.2 Trustworthiness and Confidence

In the following, we consider a network of 15 nodes, including 10 well-behaving nodes, 4 colluding nodes providing incorrect observations, and one attacker performing a link spoofing attack against a well-behaving node. The link spoofing attack engages (Figure 2.4a) and then ceases (Figure 2.4b). During the attack (Figure 2.4a), trust values of misbehaving nodes rapidly decrease regardless of their initial values and once the attack stops (Figure 2.4b), their trust values increase very slowly. This reflects the defensive nature of our trust system: it takes a lot of time to regain lost confidence while it is very easy to lose. During the attack and after, the increase of the trust associated with well-behaving is slow, especially for the well-behaved nodes that nonetheless had a small original trust value.

Rather than questioning all the neighbours, the IDS randomly selects some of the neighbours that are interogated. The diagnostic result (Figure 2.5-b) is quickly established. As expected, the confidence interval gets narrower (Figure 2.5-a) over time because more evidence are collected and the sampling error (Figure 2.6-b) drastically decreases. When the sampling error and the confidence interval are both bellow a certain threshold, it is recommended to reduce the collection of evidence, to avoid unnecessarily overloading the network (Figure 2.6).

As long as liars constitute a minority (*i.e.*, less that 50% of the nodes), trust is correctly assessed (Figure 2.6): as an illustration, attack is detected while 43.2% of the devices are lying. Attack assessment takes longer when the amount of liars increases because the trust values of the liars diminish dramatically in the last rounds. Regardless of the percentage of liars, the diagnostic result converges to -0.8.



(a) Bandwidth Usage.

(b) Impact of liars on detection.

Figure 2.6 – Bandwith usage and impact of liars.

2.5 Conclusion

In an increasingly interconnected world, an attack can be launched and massively spread from any device. Intrusion Detection Systems (IDS) hence play a crucial role on account of their ability to protect every device in a networked system including equipment, hardware and software, by forming a digital perimeter that partially or fully guards a network. We have approached the problem of intrusion detection in several contexts, spanning the routing in an ad hoc network to an RFID-based application controlling access. We adapted the IDS organisation to the contexts, leveraging an extremely distributed structure to deal with an infrastructure-less *ad hoc* network, and relying on a remote and centralised IDS with highly-contained RFIDs and an infrastructure-based IT network. We first introduced a misuse intrusion detection in which attack signatures were manually crafted in advance. We proposed an unusual approach to detect intrusions in a network, which consists in relying on the logs generated by the OLSR protocol. Logs give insight into the protocol's state and on the device view concerning the network topology, without requiring a change in implementation. After modeling the attacks and designing the corresponding attack signatures we turned our attention to network monitoring, as intrusion detection comes with awareness and continuous vigilance requirements. While monitoring is necessary, the constant surveillance of a network implies a prohibitive bandwidth consumption whereas the continuous flow of information should be regulated. To this end, we have proposed a statistical approach by leveraging the confidence level and confidence interval in order to fine-tune the evidence gathering process without sacrificing detection accuracy. In open and infrastructure-less MANETs we must also deal with the lack of faithfulness and non-corruptibility of information provided by other peers. Thus we introduced an information-theoretic trust system that evaluates trust based on previous interactions and considers recommendations given by others in order to increase the robustness of the IDS. Overall, our experiments highlighted that a signature-based intrusion detection system is suitable for tracking attacks, and in particular those targeting routing protocols in MANETs; a high detection rate and a low false positives rate are imperative. Nonetheless, anticipating the attacks and envisioning any possible inappropriate behaviours is not always practicable. In such a case, anomaly detection systems adequately complement signature-based IDS. Following, we introduced an anomaly detection system that identifies deviations from normal behaviour. First, subjects are equipped with RFID tags in order to be constantly located and monitored then, a user's profile is built, relying on Kohonen maps, which constitute an efficient method for automatically categorising and further comparing the tagged behaviour against the normal user's behaviour (as expressed in the user's profile). Our experiments [66] showed that unsupervised

machine learning used in combination with the functionality of an IDS forms a robust defence against possible attacks by predicting them based on past events. Even a simple Kohonen map, trained based on normal data, detects anomalous features that the system is exposed to. In the course of our research we observed that Kohonen maps may deal with multiple users. More features could easily be integrated leveraging vectorisation techniques. As long as the intrusion contaminates only a fraction of the training set and generates sufficiently distinct vectors, a map remains dominated by normal activity.

Chapter 3

Enhancing the Observation Quality through Calibration

3.1 Introduction

In the light of the ever expanding importance of IoT, a noticeable trend is to promote the adoption of IoT technologies, to support the sustainable development of urban spaces, thus realising the so-called vision of smart spaces and smart cities. In particular, IoT technologies contribute to the improvement of the services the city delivers and thereby respond to a number of environmental challenges generated by the city itself (*e.g.*, pollution) or caused by natural phenomena (*e.g.*, global warming). With this in prospect, various research projects, including those in which I was involved in [1, 64, 144], primarily rely on IoT infrastructures to manage and optimise some services, the oversight of public spaces (*e.g.*, campus, building, parking), environmental and pollution monitoring, surveillance for disaster preparedness and efficient first responses,as well as the preservation of cultural heritage. The delivering of problem-solving services requires the joint effort and engagement of a wide range of persons, communities, organisations and stakeholders. This also demands a(n) (r)evolution of the relationship between the participants. These needs led to the development of the crowdsensing movement wherein users (citizens, groups, communities) engage in some collaborative data collection, analysis and decision making. In practice, citizens rely on the small and low-cost sensors embedded in –or connected to– their smartphones so as to *e.g.*, help identifying environmental problems, alert about an imminent danger, collaboratively monitor road traffic, or trace the evolution of an epidemic.

Together, IoT-based sensing and mobile crowdsensing gather some observations at a fine grain, based on the fixed things of the urban infrastructure and on the mobile devices that people carry. On the one hand, stationary sensors (*e.g.*, weather stations, cameras) relay their (a priori) high-quality measurements to data sinks that redirect the resulting traffic towards the cloud (*e.g.*, a back-end server typically owned by some public organisations) – this is the typical setting involving a secured wireless sensor network connected to a wired infrastructure. On the other hand, mobile smartphones opportunistically publish their data to a (crowdsensing) server according to the connectivity allowed by their data plan. Leveraging both (mobile) individually-owned and (primarily fixed) organisation-owned sensors, smart cities may henceforth oversee a physical phenomenon at a large-scale. The resulting breed of multi-purpose applications (w.r.t apps), has shifted and keeps shifting the urban areas towards environmental, social and sustainable paths.

Nonetheless, such a vision comes with a fair share of challenges. The financial cost associated to the maintenance of the IoT infrastructure is high; limited budgets lead to a partial instrumentation of the public space. The spatio-temporal coverage problem is typically addressed by leveraging mobile crowdsensors that conveniently fill the gap thanks to the mobility of the participants. Morever, leveraging crowdsensing is a double-edged sword as crowdsensing is highly

constrained by the choice of the sensing hardware: off-the-shelf sensors/actuators outfit or are connected to the smartphones. In respect to this, the prevalence of low-cost and multi-purpose crowdsensing platforms conducts to discrepancy in the quality of observations.

To overcome this issue, a classical approach consists in calibrating sensors. Yet, we shall not solely count on the manufacturers to do so for several reasons: (i) manufacturers calibrate sensors within laboratories, under a specific range of conditions, using a controlled stimuli serving as ground-truth data; (ii) in many cases, sensors are delivered uncalibrated and are shipped along with a data-sheet that briefly sketches some generic calibration values; (iii) multi-purpose sensing devices (*e.g.*, smartphones) are not accompanied with a data-sheet nor even a mention to the manufacturer, version or brand of the sensors/actuators.

Obviously, any non calibrated sensors, whether mobile or static, need to be calibrated. In addition, calibration in the field is essential to ensure a proper operation of the sensing device, as aging, external conditions (such as solar radiation) and other factors (*e.g.*, activity of the end user) affect sensoring the measurements over time.

In situ calibration is essential to preserve data quality, yet challenging, for several reasons. Firstly, sensors are usually scattered over large areas and operate across long periods of time. Thus, they shall be checked and calibrated on a regular basis to preserve the measurements' quality. Secondly, massive amounts of sensors need to calibrate. The required calibration entails a significant upkeep of the sensors. A team of dedicated technicians needs to conduct the necessary parameterisations and the related manipulations in the field. This *modus operandi* is particularly well suited for the calibration of the IoT infrastructure. The situation is more problematic with mobile crowdsensing because participants do not hold the required expertise.

While essential, yet challenging, the calibration of (crow)sensors in the field has received little attention. All the proposed solutions [173, 171, 160, 114, 76, 31, 152, 107] perform a pairwise calibration: the measurements of the calibrated device then serve calibrating the non-calibrated device. This calibration applies with static and mobile sensors. Within a dense but very small-scale Wireless Sensor Network (WSN), the inaccurate readings of an uncalibrated sensor can be compensated using the measurements provided by a nearby (calibrated) sensor [173, 171, 160]. Such calibration eliminates the need for calibrating each sensor individually through the iterative and automatic calibration of sensor pairs. However, the calibration approach requires a dense deployment and is unpractical with large-scale deployments. With mobile sensing, a calibration process happens when a mobile sensing device meets another sensing device, either in a planned manner [114] or opportunistically [76, 31, 152, 107], depending on whether the mobile user is intentionally guided. In [114], a single mobile user, which is guided, is featured with a high-fidelity sensor, so that user calibrates static sensors. The problem then lies in finding the shortest path that the mobile user follows. With mobile calibration, it is assumed that two sensors that move freely, may opportunistically meet. In such an occurrence, the two devices sense the same phenomenon if their relative distance is inferior to a given threshold [76, 31, 152] or if they belong to the same spatial area [107], which is defined by, *e.g.*, a cell. The sequence of meetings/calibrations constitutes a so-called calibration path. The best calibration path is further assessed using the model of pairwise rendezvous introduced in [31], which represents the meetings as a matrix where a non-zero edge represents a pairwise calibration. Based on this matrix, the shortest calibration path starting at a high fidelity sensor, is selected.

In contrast to the above efforts, we take a more holistic approach to cost-effective calibration in smart spaces at scale. Our objective is to enable the effective calibration of the connected nodes, from the fixed Things that make up the IoT infrastructure to the mobile Things that people carry. To overcome such an issue, our approach is two-fold:

- We propose (§3.2) to send mobile units (*e.g.*, trained personnel) equipped with high-

quality (more expensive) and freshly-calibrated reference sensors so as to carry out calibration in the field. One can thus generate "sufficiently accurate" knowledge over time through the frequent calibration of the sensors in the field; however, this might result in increased maintenance costs! Careful planning of the calibration process is therefore essential for the cost-effective monitoring of the smart spaces – this is increasingly important as the number and size of smart spaces grow. We address cost-accuracy issues that arise in the deployment of affordable IoT systems, with the aim of developing a plan for the calibration of a large number of often inaccurate sensors in a smart space using high-integrity reference sensors that are mobile, such that (a) the deployment and operational costs for calibration are minimised, while (b) maintaining a sufficient observation accuracy from the sensor measurements. We take into consideration the presence of heterogeneous sensor types with varying calibration characteristics. We then program the calibration with respect to an observed phenomenon so as to maintain an adequate sensing accuracy while minimising the required effort from the mobile calibrators. We exploit the locality of IoT infrastructure in place – we leverage the fact that a static sensor may calibrate another nearby static sensor and thereby foster the automatic calibration of a small-scale wireless sensor network. Our proposed approach is application-aware and is able to take into account diverse sensing needs (sensor type, sensing accuracy) presented by the context at hand.

Going one step further, we capitalise on the IoT-based infrastructure whose sensors may opportunistically serve as *reference* to provide a bootstrapping calibration to the mobile crowdsensors that are passing by.

- Afterwards, we propose a distributed opportunistic calibration system (§3.3) that compensates the crowdsensor errors while alleviating the need for visiting each crowsensor to manually (re-)calibrate. Precisely, the distributed calibration system opportunistically leverages the presence of the nearby crowdsensors that monitor the same phenomenon. All the (macro-)calibration approaches introduced so far, operate either in an opportunistic or planned manner, and perform a pairwise calibration when two sensors are closeby, or meet. Instead, our solution leverages all the calibrated crowdsensors in the relevant sensing range in order to implement a multi-party calibration, which –we show– improves the performance of the calibration in terms of the resulting sensing accuracy. Our calibration system hence generalises the automated calibration to the multi-party case. Following the opportunistic calibration of crowdsensors as they meet, we introduce a *multi-hop, multi-party calibration* algorithm that is such that the history of the calibrated crowdsensors is used to assess the *best calibration hyperpath*, which is the one that minimises the accumulated calibration error. Such an opportunist calibration approach is particularly well suited to mobile crowdsensing scenarios wherein the crowd senses and meets in public place. Last but not the least,

Once calibrated, the sensors, - static and mobile – are operational and the collection of the observations accross time and space can begin.

3.2 Planned Calibration of an IoT Infrastructure

Note: This work results from a collaboration with the Distributed Systems Middleware (DSM) Group, under Professor Nalini Venkatasubramanian of the Department of Information & Computer Science at the University of California Irvine.

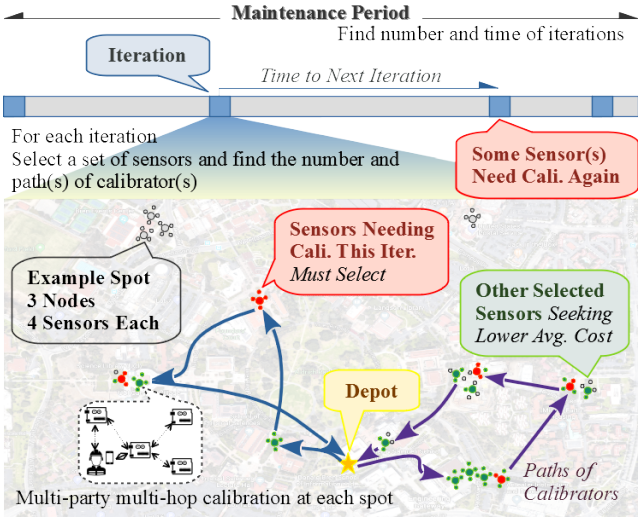


Figure 3.1 – Calibration planning over a long maintenance period.

Cost-effectively planning the on-site calibration of IoT devices helps in the sustainable long term operation of deployments. Our aim is to develop a plan (Figure 3.1) for the calibration of a large number of inexpensive sensors in a smart space using high-integrity reference sensors that are mobile, such that (a) the deployment and operational costs for calibration are minimised while (b) maintaining a sufficient observation accuracy from the sensor measurements over a given – possibly long – maintenance period T. More realistically, given the knowledge of a sensor's degradation characteristics, we program its calibration with respect to an observed phenomenon so as to maintain an adequate sensing accuracy while minimising the required effort from the mobile calibrators.

With that goal in mind, (i) we firstly partition the space to form calibration *spots* by exploiting the locality of in-situ sensors; a sensor can calibrate other sensors of the same type at the same spot. (ii) Subsequently, we carry out multiple iterations of calibration during the given maintenance period T. Our challenge lies in determining the number of iterations and the time at which each iteration should be executed. Furthermore, at each iteration we determine: the sensors to be calibrated, the number of mobile calibrators needed, and the paths taken by each mobile calibrator. The sensor calibration is then carried out accordingly over the least number of Ω **iterations** so that the overall cost over T is kept to a minimum and ensures that all the sensors always comply with the *data accuracy requirements*.

We formalise the above (Section 3.2.1) as a long-term multi-sensor calibration planning problem, which we solve using a two-phase iterative solution (Section 3.2.2) and a family of heuristic methods to enable the cost-effective planning of multi-sensor calibration in large smart spaces and over longer time periods (Section 3.2.3).

3.2.1 Multi-Sensor Calibration Planning

A **mobile calibrator** (or simply a *calibrator*) m_i is a person who carries some sensors and visits the field to calibrate the deployed low-cost sensors. Sensor calibration takes place when a calibrator m_i visits a node n_j and stays at the spot for long enough to calibrate a sensor $n_{j,k}$.

A **node** n_j, j=1, 2, ..., N, corresponds to a IoT device that embeds one or several types of low-cost **sensors**. A **sensor type** s_k, k=1, 2, ..., K, refers to the capability of detecting a certain type of phenomenon (*e.g.*, temperature, gas concentration), which usually requires a specific kind (or combination) of low-cost sensor(s). Afterwards, we denote an individual sensor

by $n_{j,k}$, and we introduce the binary **sensor presence** matrix $\mathbf{Q}_{N\times K}$ to characterise the set of available sensors so that $Q_{j,k}$=1 (resp. 0) if sensor type s_k is present (resp. absent) on node n_j.

Calibration-Related Terms

Each sensor type s_k is associated with a **calibration time** τ_k that ranges from a few seconds to several minutes, depending on the phenomenon detected by the type of sensor and the calibration complexity. We associate each sensor type s_k with a **calibration period** T_k, which characterises the maximum duration, during which the sensors remain valid (*i.e.*, the measurements have sufficient accuracy) once calibrated. The period depends on the usage scenario and may be learned from empirical studies.

During the operation, each individual sensor $n_{j,k}$ is associated with a **time to next calibration** (TTNC) $F_{j,k}$, which indicates how soon the sensor needs to be (re-)calibrated. The TTNC matrix $\mathbf{F}_{N\times K}$ then represents the TTNCs of all the sensors. $\mathbf{F}$ is a function of time, where each $F_{j,k}$ decreases between iterations and is reset to T_k when $n_{j,k}$ is calibrated. We further denote $\mathbf{F}[\omega_-]$ (resp. $\mathbf{F}[\omega_+]$), the matrix TTNC immediately before (resp. after) the iteration ω. Note that $\mathbf{F}$ is non-negative, *i.e.*, $F_{j,k}[\omega_\pm]\geqslant 0$, $\forall(j,k)$, $\omega\in\mathbf{N}_+$. If node n_j does not hold a sensor of type s_k, the corresponding TTNC is infinite.

A **sensor selection** is a collection of sensors (selected for calibration) represented by a binary matrix $\mathbf{\Gamma}_{N\times K}[\omega]$, where $\Gamma_{j,k}\leqslant Q_{j,k}$. If sensor $n_{j,k}$ is selected for calibration at iteration ω (*i.e.*, $\Gamma_{j,k}[\omega]$=1), then, at the end of this iteration, its TTNC is reset to its calibration period T_k, *i.e.*, $F_{j,k}[\omega_+]$=T_k; otherwise, its TTNC stays unchanged (during iteration ω):

$$\mathbf{F}[\omega_+] = \mathbf{\Gamma}[\omega] \circ \mathbf{T}_N + (1 - \mathbf{\Gamma}[\omega]) \circ \mathbf{F}[\omega_-] \tag{3.1}$$

Where $\mathbf{T}_N$ is the nodal calibration period matrix that consists of N identical rows of $[T_1, T_2, \cdots, T_K]$; "$\circ$" is the element-wise multiplication of matrices. Also, special needs and unexpected changes could be easily addressed by altering the TTNC matrix. Hence, the **time to next iteration** (TTNI) after ω is the minimum TTNC of all the sensors, *i.e.*, $t_{\omega+1}-t_\omega = \min \mathbf{F}[\omega_+]$; thus, the TTNC matrix immediately before the next iteration is $\mathbf{F}[(\omega+1)_-]$=$\mathbf{F}[\omega_+]-\min \mathbf{F}[\omega_+]$.

Smart-Space-Related Terms

Given the node locations $\mathbf{D}$, we can derive the **spot selection** vector $\mathbf{h}_L$ from the sensor selection $\mathbf{\Gamma}$. A spot is selected for iteration ω if any sensor on any node deployed at that spot is selected in $\mathbf{\Gamma}[\omega]$ *i.e.*,

$$h_l[\omega] = \bigvee_{j=1}^{N} \bigvee_{k=1}^{K} \Gamma_{j,k}[\omega]\cdot D_{j,l} \tag{3.2}$$

In each iteration, the **path** of a calibrator is an ordered sequence that starts from the depot and visits a set of non-repeating selected spots. It can be represented as a binary matrix $\mathbf{W}_{L\times L}$, where W_{l_1,l_2}=1 if the calibrator visits spot ν_{l_2} immediately after visiting ν_{l_1}; or 0 otherwise. The path of calibrator m_i in iteration ω is denoted $\mathbf{W}_i[\omega]$. Each selected spot is visited exactly once by one calibrator:

$$\sum_{i=1}^{M}\sum_{l=1}^{L} W_{l,l_0,i}[\omega] = h_{l_0}, \quad l_0 = 2, 3, \ldots, L \tag{3.3}$$

3.2.2 Multi-Sensor Calibration Optimisation Problem

We now introduce the **multi-sensor calibration planning** problem to minimise the **average cost** of operation over the maintenance period T. The operation cost of any iteration ω, denoted $C[\omega]$ incurs three following types of costs:

- The *iteration overhead* C_{it} is the cost related to the preparation, equipment, and the transport to the deployment.
- The *calibration cost* C_c reflects the time and effort required to conduct sensor calibration while staying at the spots. $C_c[\omega]{=}\sum_i C_{c,i}[\omega]$.
- The *movement cost* C_w is associated with the travel time of the calibrators while moving between spots. $C_w[\omega]{=}\sum_i C_{w,i}[\omega]$.

The **total operation cost** is the weighted sum of:

$$C[\omega] = \mu_0{\cdot}C_{\text{it}} + \mu_c{\cdot}C_c[\omega] + \mu_w{\cdot}C_w[\omega] \tag{3.4}$$

The **calibration cost** C_c reflects the time and effort it takes to conduct sensor calibration while staying at the spots. The cost $C_{c,i}[\omega]$ of a specific calibrator m_i in iteration ω is computed based on a given selection of sensors $\mathbf{\Gamma}[\omega]$. We further assume that the calibration that happens at the same spot is done in parallel. Thus, the **calibration time** that m_i spends at spot ν_l equals the maximum τ_k of all selected sensors at that spot (*i.e.*, $\Gamma_{j,k}{=}1$ and $D_{j,l}{=}1$). Then $C_{c,i}[\omega]$ equals the sum of the calibration times at all the spots assigned to m_i:

$$C_{c,i}[\omega] = \sum_{l=1}^{L} \left(\max_{j,k} \left(D_{j,l}{\cdot}\Gamma_{j,k}[\omega]{\cdot}\tau_k \right){\cdot} \sum_{l'=1}^{L} W_{l,l',i}[\omega] \right) \tag{3.5}$$

The **movement time** $C_{w,i}[\omega]$ of a single calibrator m_i in iteration ω can be computed from the map G and the calibrator's path $\mathbf{W}_i[\omega]$. It equals the sum of the weights on the edges between all the consecutive pairs of spots visited by the calibrator:

$$C_{w,i}[\omega] = \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \left(W_{l_1,l_2,i}[\omega]{\cdot}G_{l_1,l_2} \right) \tag{3.6}$$

Problem formulation of the multi-sensor calibration planning

Our **multi-sensor calibration planning** problem, which aims at minimising the **average cost** of operation over the maintenance period, is formulated as follows: Given the time span T, the map $\mathbf{G}$, the location matrix $\mathbf{D}$ and the sensor presence matrix $\mathbf{Q}$ of all the nodes, the calibration time τ_k and the calibration period T_k of all the sensor types, and the initial TTNC matrix $\mathbf{F}[1\_]$; find the total number of iterations Ω, and for each iteration $\omega{=}1, 2, ..., \Omega$, find the time t_ω it takes place, the sensor selection $\mathbf{\Gamma}[\omega]$, and the number and the paths of calibrations

$\{\mathbf{W}[\omega]\}$; such that the **average cost** of all iterations over the time span T is minimised:

$$\min \ \frac{1}{T} \cdot \sum_{\omega=1}^{\Omega} C(\mathbf{\Gamma}[\omega], \{\mathbf{W}[\omega]\}) \tag{3.7}$$

$$\text{s.t. } t_1 = 0$$

$$\Gamma_{j,k}[\omega] \in \{0,1\}, \ \ \forall\omega, \forall j, \forall k$$

$$W_{l_1,l_2,i}[\omega] \in \{0,1\}, \ \ \forall\omega, \forall i, \forall(l_1,l_2)$$

$$F_{j,k}[\omega_-] \geqslant 0, \ \ \forall\omega, \forall j, \forall k$$

$$F_{j,k}[\omega_+] > 0, \ \ \forall\omega, \forall j, \forall k$$

$$\Gamma_{j,k}[\omega] \leqslant Q_{j,k}, \ \ \forall\omega, \forall j, \forall k$$

$$\mathbf{F}[\omega_+] = \mathbf{\Gamma}[\omega] \circ \mathbf{T}_N + (1 - \mathbf{\Gamma}[\omega]) \circ \mathbf{F}[\omega_-], \ \ \forall\omega$$

$$\mathbf{F}[(\omega{+}1)_-] = \mathbf{F}[\omega_+] - \min \mathbf{F}[\omega_+], \ \ \forall\omega$$

$$t_{\omega+1} = t_\omega + \min \mathbf{F}[\omega_+], \ \ \forall\omega$$

$$t_\Omega + \min \mathbf{F}[\Omega_+] \geqslant T \tag{3.8}$$

$$h_l[\omega] = \bigvee_{j=1}^{N} \bigvee_{k=1}^{K} \Gamma_{j,k}[\omega] \cdot D_{j,l}, \ \ \forall\omega, \forall l \tag{3.9}$$

$$\sum_{i=1}^{M} \sum_{l=1}^{L} W_{l,l_0,i}[\omega] = h_{l_0}, \ \ l_0{=}2, 3, \ldots, L, \forall\omega \tag{3.10}$$

$$C_i(\mathbf{\Gamma}[\omega], \mathbf{W}_i[\omega]) \leqslant \hat{c}, \ \ \forall\omega, \forall i \tag{3.11}$$

$\{\mathbf{W}[\omega]\}$ are valid path(s): constraints in 3.2.3 apply.

where $C[\omega]$ is representative of the total cost of iteration ω given by Equation (3.4), which depends on the sensor selection $\mathbf{\Gamma}$ and the calibrators' paths $\{\mathbf{W}[\omega]\}$, *i.e.*, $C[\omega]{=}\, C(\mathbf{\Gamma}[\omega], \{\mathbf{W}[\omega]\})$; obviously, it also depends on problem inputs (*i.e.*, $\mathbf{G}$, $\mathbf{D}$, $\mathbf{Q}$, etc.) which are hidden for cleaner expressions. The unnumbered constraints relate to the definition of sensor selection and TTNC. Constraint (3.8) says the iterations need to cover the entire time span of T; (3.9) and (3.10) make sure all the spots with selected sensors in $\mathbf{\Gamma}$ are visited in $\{\mathbf{W}\}$; (3.11) says no calibrator should work for longer than $\hat{c}$ in any iteration. Additional constraints ensure that $\{\mathbf{W}\}$ are valid path(s).

The sensor calibration planning problem is NP-hard. It tries to minimise the total cost of all iterations while the choices of early iterations can affect and limit the choices of latter ones. Moreover, the cost of each iteration $C[\omega]$ involves a movement time $C_w[\omega]$, which also needs to be minimised, and thus requires an optimisation on the paths of the calibrators, which is a variant of the Multiple Travelling Salesman Problem (mTSP) that is known to be NP-hard.

3.2.3 Solutions and Derived Algorithms

Our formulation suggests we find $\mathbf{\Gamma}[\omega]$ (sensor selection) and a path plan $\{\mathbf{W}[\omega]\}$ (path plan) simultaneously for all iterations. However, if we know which spots the calibrators need to visit, we can optimise the paths to visit them accordingly. Hence, we attempt a two-phase local optimisation on the **single-iteration average cost**, $C[\omega]/(t_{\omega+1}{-}t_\omega)$, where we decouple the optimisation of $\mathbf{\Gamma}[\omega]$ and $\{\mathbf{W}[\omega]\}$. Accordingly, for each iteration we have a **sensor selection planning** phase and a **multi-path planning phase**. In the selection planning phase, given the initial TTNC matrix $\mathbf{F}[\omega_-]$, we optimise the sensor selection $\mathbf{\Gamma}$, from which we derive the set of selected spots H, which is then used in the path planning phase to decide the number of

calibrators and the optimal path(s) to visit the selected spots.

Sensor Selection Planning Algorithms

Leveraging the discrete nature of TTNC and the definition of TTNI (time to next iteration), we propose the TTNI-driven local optimisation algorithm. The intuition behind this algorithm is to exhaust the possible values of TTNI (*i.e.*, $t_{\omega+1}-t_\omega$) and find the "cheapest" one to fulfil. The procedure of the TTNI-driven local optimisation involves the following steps: (1) Determine **all the possible values of TTNI** that could result from any possible sensor selection in this iteration. The minimum TTNI candidate is $\min\{F_{j,k}[\omega_-] \mid Q_{j,k}{=}1 \wedge F_{j,k}[\omega_-]{>}0\}$, selecting only the sensors that need immediate calibration. The maximum TTNI candidate is $\min T_k$, selecting **all sensors**. All values in $\mathbf{F}[\omega_-]$ between them become TTNI candidates. In the worst case, the number of TTNI candidates is $O(N{\cdot}K)$ (2) For each TTNI candidate T_{cand}, tentatively assume it to be the desired TTNI and create the minimum selection of sensors to meet the TTNI, *i.e.*, let $\Gamma_{j,k}{=}1$ if $Q_{j,k}{=}1$ and $F_{j,k}[\omega_-]{<}T_{\text{cand}}$; then add all the sensors that are co-located with the selected sensors and that do not induce extra time for calibration, because their calibration is done in parallel, if it takes a shorter time). Generating $\mathbf{\Gamma}$ from T_{cand} takes $O(N{\cdot}K{+}N{\cdot}L)$ time. Compute the single-iteration average cost from $\mathbf{\Gamma}$. (3) Select the TTNI candidate that gives the minimum average cost, and its corresponding $\mathbf{\Gamma}$ is the output of the algorithm. The worst-case running time excluding the time used to compute or estimate the movement time, is $O(N^2{\cdot}K^2{+}N^2{\cdot}K{\cdot}L)$.
If during step (2) we are able to compute the optimal paths of calibrators, we will compute the best cost evaluation for each selection and find the local optima.

Multiple-Path Planning Algorithms

The objective of the multi-path planning problem is to generate a set of paths $\{\mathbf{W}[\omega]\}$ of minimum cost (*i.e.*, movement time $C_w[\omega]$) for the selected spots yielded by the sensor selection $\mathbf{\Gamma}[\omega]$. It is a variant of the classic mTSP or VRP: we determine the number of calibrators based on the demand instead of having the number m of travellers given, as in mTSP. Also, evaluating the calibrator workload constraint involves the movement time of individual calibrators, which adds to the complexity of solutions. Hence, we derive the following mixed-integer-programming (MIP) formulation of the multi-path planning problem based on a flow-based three-index MIP formulation of mTSP [19], adding appropriate modifications to match our assumptions and constraints: Given a map $\mathbf{G}$, the location of the nodes $\mathbf{D}$, the sensor selection $\mathbf{\Gamma}$, and the calibration time τ_k, $\forall k$; find $\mathbf{W}_{L\times L\times M}$ and helper variables $\mathbf{U}_{L\times M}$ to

$$\min \sum_{l_1=1}^{L}\sum_{l_2=1}^{L}\left(G_{l_1,l_2}\cdot\sum_{i=1}^{M}W_{l_1,l_2,i}\right) \tag{3.12}$$

$$\begin{aligned}
\text{s.t. } & W_{l_1,l_2,i}\in\{0,1\},\ \ \forall i,\forall(l_1,l_2)\\
& W_{l,l,i}=0,\ \ \forall l{=}2,3,\ldots,L,\forall i\\
& \textstyle\sum_{l_2=1}^{L}W_{1,l_2,i}=1,\ \ \forall i\\
& \textstyle\sum_{l_1=1}^{L}W_{l_1,l,i}-\sum_{l_2=1}^{L}W_{l,l_2,i}=0,\ \ \forall i,\forall l\\
& \textstyle\sum_{l_1=1}^{L}\sum_{i=1}^{L}W_{l_1,l_2,i}=h_l,\ \ \forall l_2
\end{aligned} \tag{3.13}$$

$$u_{l,i}\geqslant 2,\ \ \forall i,\forall l \tag{3.14}$$

$$u_{l_1,i} - u_{l_1,i} + 1 - (L-1)\cdot(1 - W_{l_1,l_2,i}) \leqslant 0, \forall i, \forall (l_1, l_2)$$
$$\sum_{l_1=1}^{L} \sum_{l_2=1}^{L} W_{l_1,l_2,i}\cdot\Big(G_{l_1,l_2} + \Upsilon_{l_2}\Big) \leqslant \hat{c}, \ \ \forall i \tag{3.15}$$

Where $W_{l_1,l_2,i}$=1 if calibrator m_i visits spot ν_{l_2} immediately after spot ν_{l_1}; or 0 otherwise. M is the maximum number of calibrators; assuming we always have enough calibrators, L would be an effective upper bound of M to be used in solvers. The unnumbered constraints are related to the construction of multiple paths. Constraint (3.13) makes sure all selected spots are visited by exactly one calibrator; (3.15) enforces the maximum workload of calibrators, where Υ_l is the total calibration time spent at spot ν_l, *i.e.*, $\Upsilon_l[\omega] = \max_{j,k}(D_{j,l}\cdot\Gamma_{j,k}[\omega]\cdot\tau_k)$.

However, the problem is NP-hard; the number of independent variables and the number of constraints in this MIP formulation are both in the order of $O(L^3)$, resulting in a huge solution space. It is hard for any MIP solver to optimally solve the problem in a reasonable amount of time [19]. In practice, we tried two widely used solvers: GLPK (GNU Linear Programming Kit, open-source – https://www.gnu.org/software/glpk/) and Gurobi (commercial software – http://www.gurobi.com/). None of the two solved the problem in less than 48 hours for $L \geqslant 15$. To solve the problem at a larger scale, we propose two heuristics: a greedy algorithm derived from the nearest neighbour heuristic of traditional TSP, and an improved genetic algorithm (GA) based on the one proposed by Sedighpour, et al. [154] for mTSP. For a clean design of the algorithms, after the completion of the sensor selection planning phase, the planning framework computes the set of **selected spots** H and the calibration time $\boldsymbol{\beta}$ at these spots from the sensor selection matrix $\boldsymbol{\Gamma}$.

Nearest-Neighbor-Based Greedy Heuristic

The nearest neighbor algorithm for TSP starts with a tour containing only one spot. At each step, it determines the next spot to visit as the one that is closest to the last visited spot, and loops until all the spots are visited. Inspired by this straightforward TSP algorithm, we derive that our greedy algorithm (see Algorithm 1) for the multi-path planning problem, works as follows: (1) Start with a set of empty paths (*i.e.*, all the calibrators stay at the depot) and the set of all selected spots H. (2) At each step, for each unvisited spot, pick the spot-calibrator pair that induces the least additional movement time. (3) Loop until all the spots are visited. Note: Our actual implementation of this algorithm caches the movement and calibration time associated with each calibrator to reduce redundant computation, so the worst-case running time of this algorithm is $O(L^3)$.

Improved Genetic Algorithm (GA)

We design our genetic algorithm (GA) based on the mTSP GA solution of [154]. Features are added to address the peculiarities of our MPP formulation, *i.e.*, the variable number of calibrators, the workload constraint, and the map (*i.e.*, a directed-graph).

A **chromosome** is an integer vector that is made of two parts: a permutation of all the selected spots (1st half) and an assignment mapping the spots to mobile calibrators (2nd half). If the number of selected spots is $|H|$=L', a chromosome's length will be of $2L'$. The assignment (2nd half) is represented by the number of spots visited by each calibrator, so these integers should all be in range $[0, L']$ and sum up to L'. For example, chromosome $[2, 4, 5, 6, 3, 2, 3, 0, 0, 0]$ means L'=5 and that m_1 will visit spots ν_2, ν_4, and m_2 will visit ν_5, ν_6, ν_3. The **fitness** is the negative of the total movement time of all the mobile calibrators, and the **selection** is done by a standard scaled-fitness proportional selection.

The **initial population** is composed of randomly generated individuals. The permutation is performed by a uniformly random permutation generator, and the assignment is done by uniformly and randomly picking an integer and subtracting it from the total number of selected spots until none is left. The **crossover** is done by applying a standard "order crossover" on the first half of the chromosome.

Because of the variable number of mobile calibrators, we design three helper functions that apply to chromosomes: (a) **compress**: shift all zeros in the assignment section to the end and non-zeros values to the beginning; (b) **split**: check if any assignment ($\geqslant$2 spots) leads to an overloaded calibrator, randomly split it into two calibrators, and loop until none is found; (c) **merge**: check if there exists a pair of assignments that can be merged into one without overloading the calibrator; then merge the first pair found. Among the three, **compress** and **split** are applied to every newly-generated chromosome during population initialisation, mutation, and crossover, while **merge** is applied as one type of mutation.

Apart from "merge", there are three other types of **mutation**: (a) two-point swap, (b) segment reversal, and (c) 3-opt local optimisation. Every time a mutation is triggered, we randomly pick one of the four types of mutation functions. (a) and (b) are straightforward. 2-opt is a commonly used local optimisation for TSP on undirected graphs, but an odd numbered opt is required for digraphs to avoid reversing any segment, which makes it faster to compute the new movement time.

Finally, the tunable parameters such as the population size, elite-keeping size, and the termination conditions are assigned by the framework according to the problem size (*i.e.*, L').

Overall, we have presented a collaborative and distributed calibration solution that enhances the quality of the gathered observations. We frame multi-sensor calibration planning as an optimisation problem where we propose a two-phase iterative local optimisation approach to determine (a) how many calibration iterations are necessary, (b) which sensors should be calibrated at each of these iterations, and (c) the number of mobile calibrators that are required (as well as their respective calibration paths). Such an effective deployment and maintenance of the IoT infrastructure within a metropolitan area fosters the long term gathering of meaningful observations. In addition, calibrated IoT infrastructure can in turn serve as a high-quality reference to opportunistically calibrate some mobile sensors that monitor the same phenomenon and that are engaged to sense at a larger scale.

3.3 Robust Multi-Party calibration

The distributed collaborative calibration requires rendezvous, although infrequent, with reference sensor(s) belonging to the IoT infrastructure, for the calibration of the mobile crowdsensors. An overriding requirement is to minimise the involvement of end-users, who usually do not hold the required expertise. The calibration system shall require no specific knowledge or manual configuration – end-users do not need to place themselves under ideal conditions. End-users engaged in the crowdsensing, are simply required to install a mobile app – implementing the calibration and the crowdsensing together– on the smartphones at their disposal.

Our solution leverages robust regression so as to discard the measurements that are of too low quality for being meaningful. We further introduce a multi-party calibration system, which exploits the spatial dimension of the problem which –we show– improves the performance of the calibration in terms of the resulting sensing accuracy. Several sensors that are separated in space, may track better the physical phenomenon and should therefore be involved into a multi-party calibration of higher-quality. In particular, we say that there is a *multi-party calibration rendezvous* between a non-calibrated smartphone i and m calibrated smartphones

$(1 \leq m \leq n-1)$ in the case of both spatial and temporal proximity over a time period. That is, if the following conditions are met:

- The m reference – or previously calibrated– sensors are in communication range with i during the overall calibration period $\{t_1, ..., t_p\}$.
- The m reference sensors are in a *shared sensing range* with i, *i.e.*,, the distance between i and any reference sensor $j \in \{1, \cdots, m\}$ is less than the maximum calibration distance d_{max}, which is defined based on the measured physical phenomenon and environment being considered.

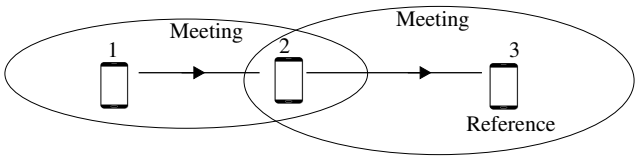
During the multi-party rendezvous, the uncalibrated sensor i attempts to calibrate using the calibrated measurements provided by the m nearby sensors. The objective lies in determining a calibration function, which describes the relationship between the non-calibrated measurements and the calibrated one(s) and thereby enables correcting the measurements of the non-calibrated sensor(s). As presented in literature, for many sensors, there is close to linear dependence between the measurements produced by the sensor and the sensed phenomenon (*e.g.*, see [105, 30]). Accordingly, it is assumed that a sensor i can be provided with the relevant calibration coefficients/parameters for the linear function, which returns the calibrated measurements $\hat{y}_i$ given the sensor measurements $\hat{y}_i$ at time t. In the same way, the measurements of a non-calibrated sensor have a linear dependence with those of some calibrated sensors, as illustrated in Figure 3.3 with the specific example of noise sensing using the smartphones' microphones. In this case, any reading $y_i(t)$ of the uncalibrated sensors (at time t) can be expressed as the following function: $y_i(t) = \beta_0 + \beta_1 x_1(t) + \beta_2 x_2(t)$ with $x_1(t)$ and $x_2(t)$ corresponding to the reading of two calibrated sensors. Once calibrated, the sensor i can in turn, be used to calibrate others. Such a dynamic calibration, which is also known as *blind calibration*, allows calibrating sensors under their deployment conditions and without involving the end-user.

To the best of our knowledge, this is the first calibration system that generalises the automated calibration to the multi-party case. Following the opportunistic calibration of sensors as they meet, we introduce a *multi-hop, multi-party calibration* algorithm that is such that the history of the calibrated sensors is used to assess the *best calibration hyperpath*, which is the one that minimises the accumulated calibration error. Figure 3.2a depicts the pairwise and multi-hop case supported in literature [76, 31, 152], which is based on a sequence of pairwise calibration rendezvous with the initial calibration starting with the Reference sensor. Figure 3.2b then introduces the general case of multi-hop, multiparty calibration, in which a succession of pairwise or multi-party calibrations may take place. In practice, our sensor calibration aims at assessing the sensor's bias compared to the measurements of reference (or previously calibrated) sensor(s) that happen to be in the relevant communication and sensing range. Overall, the calibration system is unobtrusive in that it requires no configuration or handling once started. Furthermore, the system does not require a parsimonious and dense deployment, making it practically relevant to various urban scenarios.

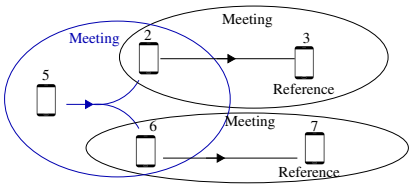
3.3.1 Multivariate Linear Regression

Given the multi-party calibration rendezvous of the sensor i with m smartphones, the *multivariate linear regression* computes the calibration coefficients/parameters for i based on the calibrated measurements provided by the m mobile sensors. In particular, the uncalibrated (raw) measurement of the sensor i is defined as the following linear function:

$$Y_i = XB + \mathcal{E}_i \tag{3.16}$$



(a) *Multi-hop pairwise calibration:* Sensor 2 first calibrates with the reference sensor 3. Then, sensor 1 may calibrate with calibrated sensor 2.



(b) *Multi-hop pairwise followed by multiparty calibration:* Sensors 2 and 6 apply pairwise calibration. Then, multiparty calibration occurs for sensor 5.

Figure 3.2 – Pairwise versus multi-party calibration.

Y_i is a p-dimensional vector (*i.e.*,, $Y_i = (y_i(t_1), \cdots, y_i(t_p))^T$) with T denoting the transpose of the vector; X is a $p \times (m+1)$ that includes the calibrated measurements of the m sensors ($j \in \{1, \cdots, m\}$); B is a $(m+1)$-dimensional vector (*i.e.*,, $B = (\beta_0, \cdots, \beta_m)^T$) that represents the unknown (and fixed) regression coefficients; and and $\mathcal{E}_i$ is a p-dimensional vector (*i.e.*,, $\mathcal{E}_i = (\varepsilon_i(t_1), \cdots, \varepsilon_i(t_p))^T$), which refers to the residual noise. The model assumes that $\mathcal{E}_i$ is a normally distributed random variable with 0 mean and a constant standard deviation σ that is unknown [49]. Given the above, we want to find the regression coefficients B of $\hat{Y}_i$:

$$\hat{Y}_i = XB \tag{3.17}$$

We apply the method of the least square, which minimises the sum of the squared differences between the actual values y_i (resp. Y_i) and the regression values $\hat{y}_i$ (resp. $\hat{Y}_i$), using the linear regression of Equation 3.16. Formally, the estimate of the regression coefficient denoted $B = (\beta_0, \cdots, \beta_m)^T$ is computed such that $\sum_{t=t_1}^{t_p}(y_i(t) - \hat{y}_i(t))^2$ is minimised. This sum of the squared differences is minimised analytically by setting: $B = (X^T X)^{-1} X^T Y_i$. It follows that the fitted value $\hat{Y}$ verifies:

$$\hat{Y}_i = XB = X(X^T X)^{-1} X^T Y_i \tag{3.18}$$

Thus, the minimised residual error is given by:

$$\mathcal{E}_i = Y_i - \hat{Y}_i = (I - H)Y_i = (I - X(X^T X)^{-1} X^T)Y_i \tag{3.19}$$

where I is a $m \times m$ identity matrix.

The above simple multivariate regression is particularly well suited to establish a relationship between the measurements of sensor i and those of provided by the smartphones that are encountered during a multi-party calibration rendezvous. Nonetheless, the validity of the simple multivariate regression is compromised if some measurements contain outlier(s). With sensors, outliers are commonly observed.

Removing Outliers using Robust Regression

Rather than minimising the sum of the squared residuals $\sum_{t=t_1}^{t_p} \varepsilon_i(t)^2$ as it is the case with simple regressions, the *robust regression* minimises the median of squared residuals computed for the entire data set [136]. This median-based estimator resists the effect of nearly 50% of the data contaminated by outliers. Afterwards, we remove outliers using the *robust regression* algorithm presented in [136], which iterates on the following steps: Randomly select few samples; Fit the model to these few samples using multivariate regression; Evaluate the quality of the fit on the remaining points using the median, which is a well known robust estimator. Ultimately, the

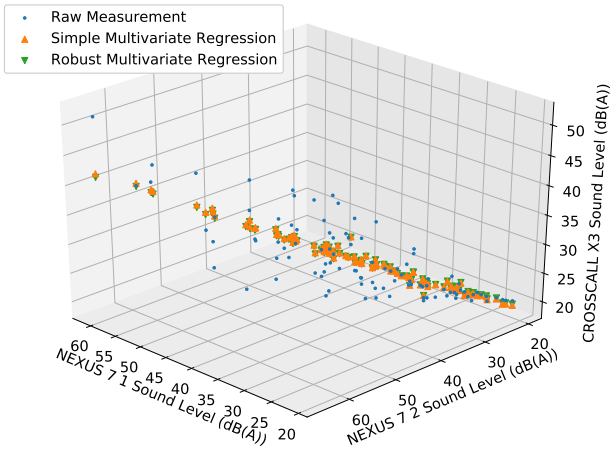


Figure 3.3 – *Simple versus robust linear regression:* Sound Level (dB(A)) of an uncalibrated Crosscall Trekker-X3 expressed as a function of the sound level sensed with two calibrated Asus Nexus 7.

subset that is characterised by the best fit is kept. Ideally, the above procedure should repeat for all possible sub samples of size q, for which there are C_p^q possible combinations. In practice, one should select a certain number of random subsamples, such that the probability that at least one of the q subsamples is good, is almost 1. A subsample is said to be "good" if it contains up to a fraction f of bad observations. The probability that at least one of the q subsamples is good, assuming that p/q is large, is $1-(1-(1-f)^q)^p$ By setting a probability of almost 1, the value of q can be determined based on the given values of p and f. Finally, the model is re-fitted to all the data that are sufficiently close to the model and the data that are not close enough (*i.e.*,, the outliers) are removed.

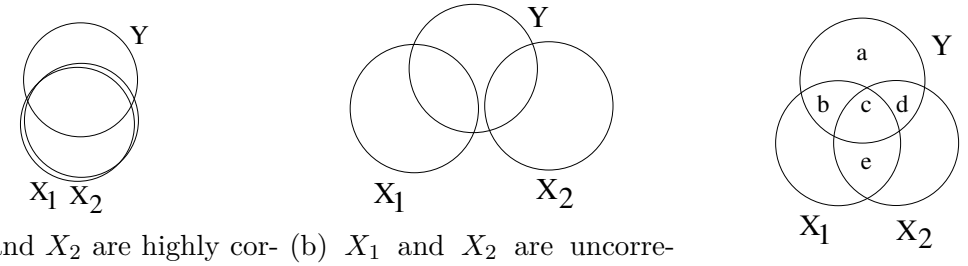
Using such a robust regression, the calibration coefficients of an uncalibrated smartphone i are estimated based on the possibly faulty readings $X_1, \cdots, X_m$ provided by m sensors that are met during the multi-party rendezvous.

Assessing the Relevance of a Calibration Rendezvous

Given a set M of m devices meeting with i, the multi-party calibration requires determining the *best subset* K of k ($k \leq m$) devices to calibrate with, *i.e.*, the subset of devices with which to exchange measurements so as to enhance the accuracy of i's measurements while minimising the resulting resource consumption. Figure 3.4 illustrates the representative cases that arise where Y (resp. X_1, X_2) depicts the variance of the samples from device i (resp. 1, 2):

- If two nearby devices, 1 and 2, supply almost the same samples (Fig. 3.4a), the calibration of the device i does not require the readings of 1 and 2; using the samples of one of the two is sufficient.
- Contrariwise, when two devices provide uncorrelated samples (Fig. 3.4b), each separate device makes a unique contribution to the calibration of i and should be considered during the calibration.

Overall, the best subset K should include the nearby devices that: (i) are both calibrated and (ii) both contribute sufficiently to the calibration of i by supplying some samples that are mostly non-redundant. We rely on (semi-)partial correlation [49] to estimate the unique contribution



(a) X_1 and X_2 are highly correlated.

(b) X_1 and X_2 are uncorrelated.

(c) X_1 and X_2 are correlated.

Figure 3.4 – Device i meets two calibrated devices 1 and 2. Device i collects samples locally, whose variance is Y and determines if it can calibrate using the samples of 1 and 2 (with variance denoted X_1 and X_2). The figure depicts the relationship between the 3 variables Y, X_1 and X_2. In 3.4c: a is the variance specific to the samples of i; b (resp. d) is the part of Y variance uniquely influenced by X_1 (resp. X_2); e is the part of X_1 variance uniquely influenced by X_2; c is the part of Y variance influenced by both X_1 and X_2.

of each device to the calibration of i. Consider the case depicted in Figure 3.4c, the area a represents the variance specific to Y, while the area $b + c + d$ (*i.e.*, the area of Y intersecting with X_1 and X_2) represents the overlap of Y with X_1 and X_2. We may further decompose the variance Y with respect to X_1 and X_2. The area b (resp. d) is the part of the Y variance uniquely influenced by X_1 (resp. X_2); it is represented by the squared semi-partial correlation coefficient, denoted r_{X_1} (resp. r_{X_2}), which reflects the correlation between Y and X_1 while X_2 has been partialed out (*i.e.*, the effect of X_2 has been removed from X_1 but not from Y). The semi-partial correlation of X_1 with Y, which can be simply denoted sr_{X_1}, corresponds to the correlation $r_{X_1 \setminus X_2 Y}$ between X_1 and Y, removing the effect of X_2 from X_1, but not from Y:

$$sr_{X_1} = r_{X_1 \setminus X_2 Y} = \frac{r_{X_1\ Y} - r_{X_1\ X_2} r_{X_2\ Y}}{\sqrt{1 - r^2_{X_1\ X_2}}} \tag{3.20}$$

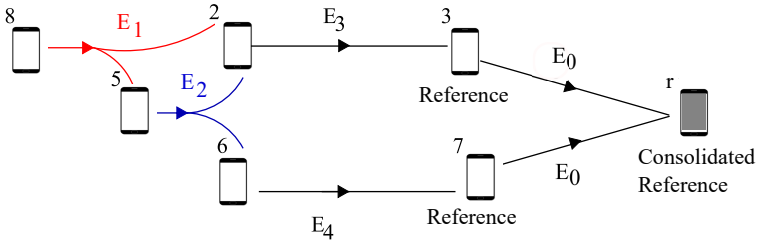
Following, we use the semi-partial correlation to characterise the specific effect that each calibrated device of M may have on the calibration of i so as to exclude any device j that contributes measurements that are not sufficient/unique (*i.e.*, with $sr_{Xj} \leq \epsilon$). We apply a stepwise process, wherein the devices having the lowest semi-partial regression are removed first. Semi-partial correlations are further used in a stepwise regression procedure, where the device (rather than the end-user) determines which variables (*i.e.*, device) should be considered in the final regression. We note that, in a forward stepwise regression, the calibrated device that would add the largest semi-partial correlation is added next (provided it is statistically significant).

3.3.2 Multi-hop, Multi-party Calibration

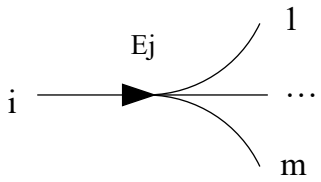
We now consider the history of the sensors' calibrations to assess the relevance of a given multi-party rendezvous. Indeed, we must compare the quality of the calibration parameters computed in the current rendezvous to the quality of the previous calibrations, if any.

Formalising the History of Multi-party Calibrations using Hypergraphs

Formally, we represent the history of multi-party calibrations using a weighted, directed hypergraph (Figure 3.5a). A *directed hypergraph* is a pair $H = (V, \mathbb{E})$, where V denotes the finite



(a) Hypergraph example. *Node* r abstracting all the *reference nodes*, to which they each connect, is added to the hypergraph.



(b) Hyperedge between node i and m nodes.

Figure 3.5 – Hypergraph encoding the multi-party calibration history.

set of nodes (*i.e.*,, sensors/smartphones) and $\mathbb{E}$ the finite set of directed hyperedges. A *directed hyperedge* $E_j \in \mathbb{E}$ is an ordered pair $E_j = (T, D)$ of two (possibly empty) disjoint subsets $T, D \subset V$, where T is the *tail* –noted $tail(E_j)$– and D is the *head* –noted $head(E_j)$– of the hyperedge. As an illustration, the *hyperedge* $E_j = (\{i\}, \{1, \cdots, m\})$ represents in Figure 3.5b the multi-party rendezvous between node i and m nodes; the cardinality of the tail is 1 (*i.e.*,, $|tail(E_j)| = 1$) while the cardinality of the head is m ($|head(E_j)| = m$, with $m \geq 1$). The directed hyperedge is further weighted such that each of the inner-edges between i and one of the m nodes provides the properties of the calibration (including regression) established by i using the readings provided by the calibrated node(s). Thus, $V^2 \mapsto \mathbb{R}^q$ denotes the set of all the inner-edges such that the weight of any $i, j \in V^2$ is defined as a q-dimensional vector that specifies:

- The characteristics of the meeting, which includes, *e.g.*, the time period during which the calibration is taking place;
- The properties of the calibration for sensor i, which is established based on the readings provided by sensor j. This includes $\hat{y}_i$ as well as the parameters that determine the quality of the regression, including the variance-covariance matrix (as previously defined) and the regression error.

Consider the hypergraph that is locally maintained by a participant node i involved in some rendezvous, *i.e.*, the hypergraph at i represents the history of past rendezvous together with the rendezvous that are currently eligible for the calibration of i. The m nodes in the sensing range of i similarly maintain their hypergraph of calibration rendezvous. The various nodes may then exchange their respective hypergraphs, which allows each one of them to compute the *shortest hyperpath* that minimises their own calibration error (*i.e.*, as estimated by the regression qualities that weight the inner edges). Consider the example of Figure 3.5a, in which there are two hyperedges from node 8 to reference nodes 3 and 7: (i) From 8 to 3 through E_1, E_2, and E_3, and (ii) From 8 to 7 through E_1, E_2, and E_4. We simplify the problem of establishing the hyperpath of minimum weight from i to a set of *reference nodes* (*i.e.*, highly accurate, high-quality sensors), denoted R, by transforming it into the problem of finding a hyperpath of minimal weight from i to a single node r. This is done by adding a new *consolidated* node r to the hypergraph and connecting each end reference node to node r with a $< 0 >$-weighted hyperedge.

Following, given a directed and weighted hypergraph $H = (V, \mathbb{E})$ and the nodes i and r of V, we say that there is an *hyperpath* from i to r shall one of the following conditions be met:

1. $i = r$, in which case, the hyperpath is empty.

2. There is a hyperedge $(\{i\}, K) \in \mathbb{E}$ such that $\forall k \in K \subset V$, there is a hyperpath from k to r.

Condition 2 signifies that a node i can calibrate using the surrounding sensors in K if and only if the sensors in K are also calibrated (*i.e.*, there is a hyperpath from each of them to r). Overall, considering the set of eligible hyperpaths, $H_{i,r} = (V_{i,r}, \mathbb{E}_{i,r})$, from i, according to the set of nodes in the sensing and communication ranges of i, the goal of node i, which attempts to calibrate, is to find the calibration hyperpath (from i to r) –and thus the associated hypergraph– that minimises the accumulated weight, which is:

$$\boldsymbol{P} : \min_{\mathbb{E}_{i,r} \subseteq \mathbb{E}} \sum_{E_j \in \mathbb{E}_{i,r}} w(E_j) h(E_j) \tag{3.21}$$

where $w(E_j)$ specifies the sum of the weights of the inner edges of an hyperedge E_j, and $h(E_j) = 1$ if the hyperpath goes through the edge E_j and $h(E_j) = 0$ otherwise.

Finding the Shortest Hyperpath

Using the local hypergraph, any sensor i may independently compute the shortest hyperpath (in terms of the edge weights) towards the "consolidated end reference" sensor r, for which we introduce Algorithm 2:

1. The initialisation (lines 2-4) is as follows: the hyperpath from i to any node is set to empty; the distance to the source is set to 0 and to all the other nodes to infinite; and the set of nodes that need to be visited, denoted Q, is set to V.

2. Then, all the nodes are visited (line 5), starting with i as the current node (*i.e.*, $u = i$). In the consecutive rounds, the current node refers to *the closest node*, *i.e.*, the closest node that is reachable by an outgoing hyperedge (line 6).

3. The set of nodes that can be reached from the current node through any outgoing hyperedge E_j is determined (line 7):

   - First, we treat the pairwise rendezvous (lines 8-14), which is characterised by a hyperedge E_j having a head of cardinality 1. If a shorter path passing through E_j is found, the distance from i to the newly connected node v is updated considering the minimal distance, and the hyperpath to v is updated accordingly.
   - Then, we treat the multi-party rendezvous (lines 15-25) with a hyperedge E_j with $|head(E_j)| > 1$, which has not been visited so far (*i.e.*, $E_j \notin E$). We establish the shortest hyperpath from each node v participating to the rendezvous (for each $v \in head(E_j)$. We check if each node v is already calibrated, *i.e.*, if there exists a hyperpath for each node v to r (*i.e.*, $H_{v,r} \neq \emptyset$) or if node r is reached. In such a case, if the hyperpath going through each v is the shortest one (so far), this hyperpath is established as the shortest one.

4. After having visited any node of the hypergraph, the shortest hyperpath established so far (if any) is returned.

The algorithm runs in $O(log\ \frac{V^{E_u+V}}{(V-E_p)^{(V-E_p)}})$ time, assuming that there are V nodes in the hypergraph (including the consolidated node r that has been added) and that there are E hyperedges with: E_u hyperedges reflecting each a single party meeting, and E_p hyperedges

designating multi-party meetings (*i.e.*, $E = E_u + E_p$). More precisely, the main loop of the algorithm extracts one node that has not been visited. All these –unvisited yet– nodes are stored in a min heap of size V which contains the node identification along with the distance to i. So, the extraction of the closest node v not yet visited (as performed in line 6) takes $O(log\ V)$. The next step consists in checking all the outgoing (single-party and then multi-party) hyperedges of v. First, we consider any single-party hyperedge and we store the shortest distance to the node v accessible through this hyperedge. If that node is not yet visited, we add it to the list of unvisited node Q; given that at most V nodes are not yet visited, we have a total cost of at most $O(VlogV)$. On the other hand, if the node v is already in Q, its priority may be increased considering the shortest distance to i, leading to a cost of $O(E_u log\ V)$. Therefore, so far, the total run time is $O(V\ log\ V + E_u log\ V)$, which is $O(E_u logV)$ because V is $O(E_u)$ assuming a connected hypergraph. In addition, we consider the analyses of E_p multi-party rendezvous. For each of them, there are at most V hyperpaths established using a hypergraph containing at most $V-1$ nodes and $E_p - 1 + E_u$, leading to the overall time complexity:

$$E_u log\ V + \sum_{i=1}^{E_p} V^i log(V-i) \sim E_u log\ V + log \prod_{i=1}^{E_p} (V-i)^{V^i} = E_u log\ V + V \sum_{i=V}^{V-E_p} log\ i! =$$
$$E_u log\ V + V \sum_{i=1}^{V-E_p} log\ i! - V \sum_{i=1}^{V-1} log\ i! < E_u log\ V + V log\ V - (V-E_p) log(V-E_p) =$$
$$log\ \frac{V^{E_u+V}}{(V-E_p)^{(V-E_p)}}$$

The multi-hop, multi-party calibration in practice

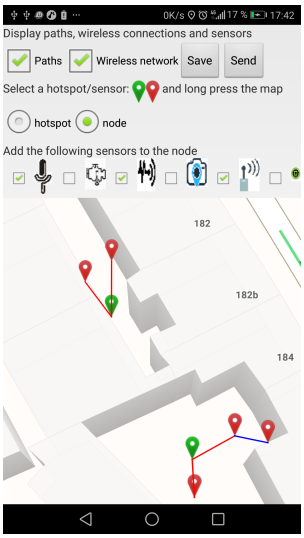
Summarising the multi-hop, multi-party calibration, any mobile i participating in the collaborative calibration periodically runs the following process:

1. Sensor i detects the presence of the nearby sensing device(s), *i.e.*, the devices in the shared sensing range.
2. Sensor i and the surrounding devices exchange their respective hypergraphs and any node updates its hypergraph (see Algorithm 2).
3. In the presence of any eligible rendezvous, i exchanges its sensing measurements (*i.e.*, time series) in a synchronised manner so as to establish the linear relationship between the measurements of the nearby sensors and the raw measurements obtained locally.
4. The best calibration path is determined, the calibration function is set and the hypergraph is updated accordingly.

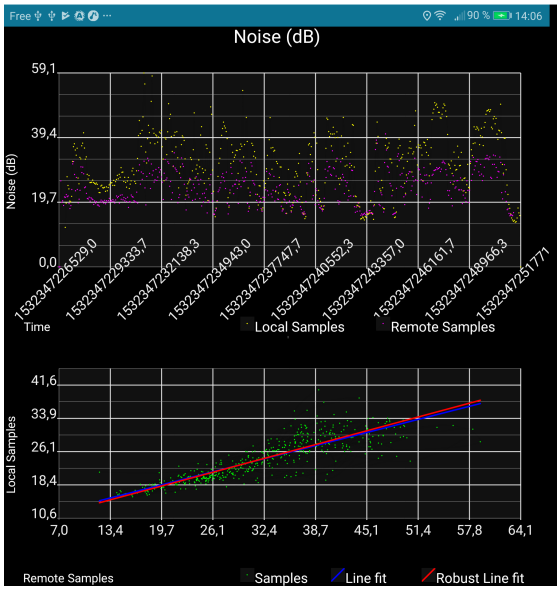
The calibration function is a linear function from the actually measured values to the calibrated values. Let the expected calibrated measurements of sensor i be $\tilde{Y}_i$. In order to calibrate the actual measurements $Y_i = (y_i(t_1), \cdots, y_i(t_p))^T$, it is required to apply the reversion of the linear relationship for: $Y_i \mapsto \tilde{Y}_i$ simply by replacing the reference values with $\tilde{Y}_i$ and replacing the regressed values $\hat{Y}_i$ with Y_i, in Equation (3.17), where $\hat{Y}_i$ eliminates the residual noise error while maintaining the linear relationship from reference to actual measurements. Thus, the calibration function is:

$$\tilde{y}_i(t) = \frac{y_i(t) - \beta_0}{\sum_{j=1}^{k} \beta_j} \tag{3.22}$$

where the index j indicates a vertex from the k reference vertices. The multi-hop allows a recursive loop for the calibration process from the reference vertices to all uncalibrated vertices.



(a) *Mapping of a space:* Spots (green markers) are placed; nodes (red markers) compose a spot and communicate via wireless links (blue lines).



(b) *Visualising a pairwise calibration:* On the top, the noise level collected by the two sensors. Bellow, sound level of the uncalibrated device is expressed as a function of the calibrated device; simple regression line (pink line) and robust regression line (blue line) are traced.

Figure 3.6 – Prototype GUI.

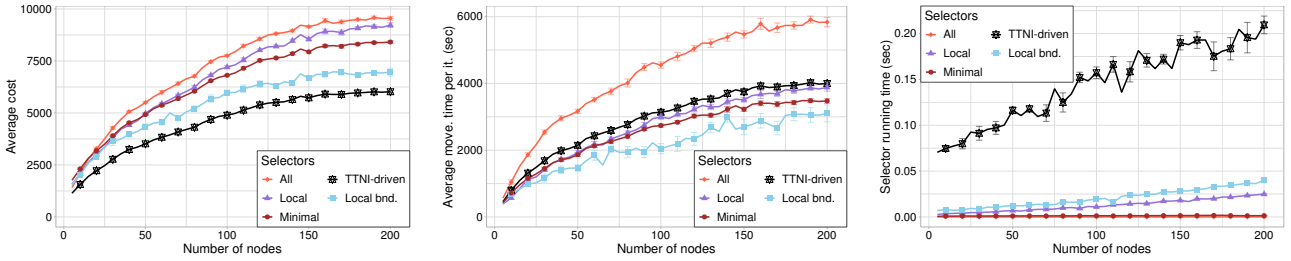
3.4 Evaluation

In order to assess our calibration solutions, we prototyped a calibration planner as well as the multi-party and the multi-hop calibrator. The calibration planer implements the planing algorithms (§ 3.2.3) and is accompanied with a service (Figure 3.6a) that serves to visualise 3-dimensional maps, place on the maps, the spots that should be visited along with the nodes and the related sensors, and besides provide navigation guidance to the mobile calibrators. In addition, we have implemented an opportunistic calibration system as an Android Application Package (APK). Figure 3.6b illustrates the GUI of the application, which plots the measurements and the regression line. The system is intended to enable pervasive calibration in a fully decentralised way, and thus does not require any central authority. While our approach supports the automated calibration of diverse sensor types, we have focused on noise sensing for our prototype and experiment.

Hereinafter, we evaluate the performance of our proposed multi-sensor calibration planner (§ 3.4.1) and of our mobile calibration system (§3.4.2).

3.4.1 Planning the Calibration

We evaluate the performance of our proposed multi-sensor calibration planner, considering the IoT structure, which is deployed at UC Irvine [21, 101] within a smart instrumented building for everyday monitoring and for emergency responses. In particular, we consider the multi-sensor platforms (called SCALE boxes) consisting of a Raspberry-Pi-based multi-sensor box interfaced with a wide range of sensor types (gas, light, air quality, temperature, seismic, camera, etc.). We conduct a series of evaluations using three sets of input data. The first two ones involve

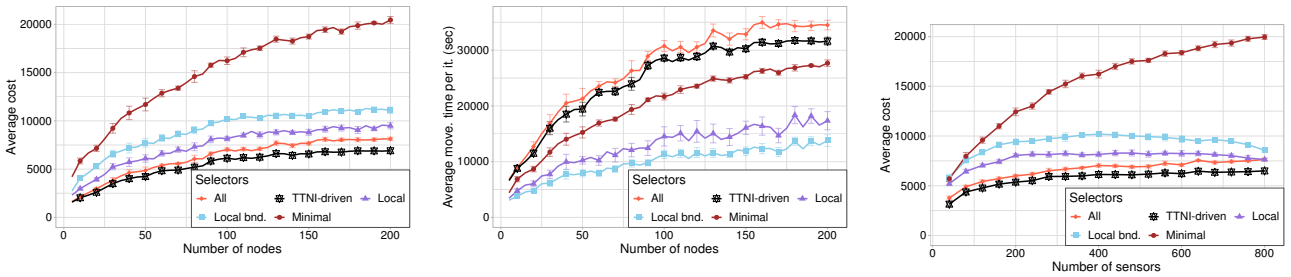


(a) Average cost vs. number of nodes.

(b) Avg. move. time (s) per iter. vs. numb. of nodes.

(c) Mean running time (sec) vs. number of nodes.

Figure 3.7 – Impact of the number of nodes on the sensor selection algorithms in an indoor setup, with sensor presence rate $\sum \mathbf{Q}/(N{\cdot}K)$=0.5.

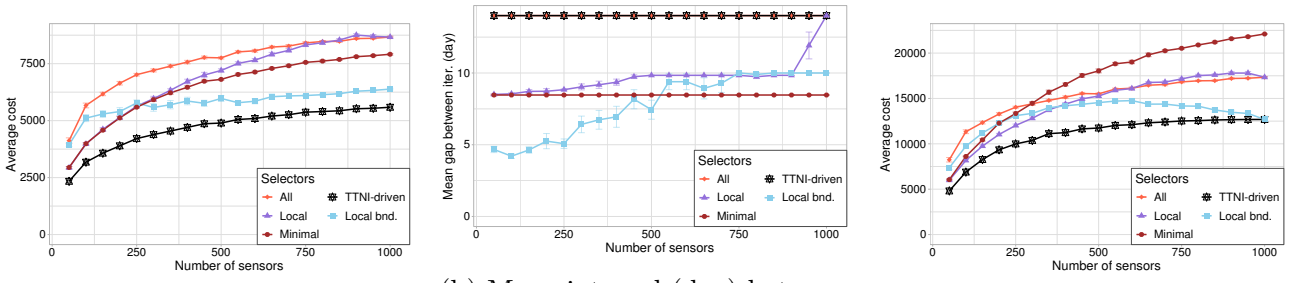


(a) Average cost vs. numb. of sensors N; sensor presence $\sum \mathbf{Q}/(N{\cdot}K)$=0.5.

(b) Average movement time (sec) per iteration vs. N; $\sum \mathbf{Q}/(N{\cdot}K)$=0.5.

(c) Average cost vs. number of sensors $\sum \mathbf{Q}$; N=100.

Figure 3.8 – Evaluation of the sensor selection algorithms; outdoor scenario.



(a) Average cost vs. number of sensors; normal scenario.

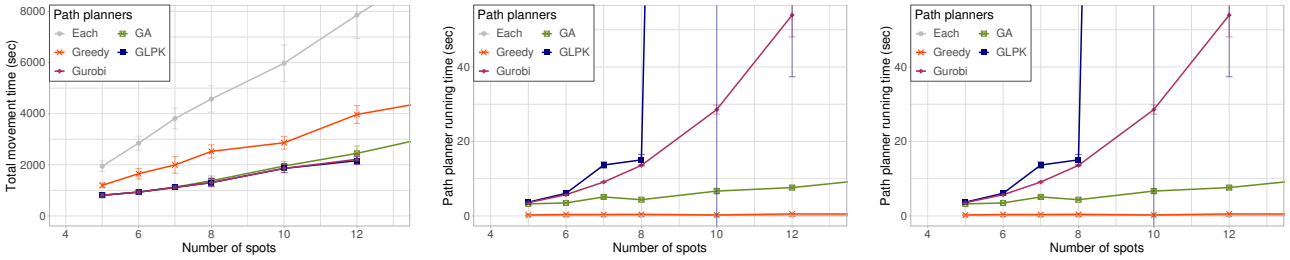
(b) Mean interval (day) between iterations vs. number of sensors; normal condition.

(c) Average cost vs. number of sensors; emergency condition.

Figure 3.9 – Impact of the number of sensors on the sensor selection in an indoor environment containing 100 nodes.

the instrumented building at UC Irvine: we consider the actual deployment and we generate additional spots (with sensors and nodes) using a similar pattern. We also synthesised the third data set that relates to an outdoor urban environment in which sensors are placed to monitor noise and air quality. The parameters are summarised in Table 3.1.

In the following, we compare our sensor selection with two naïve sensor selection strategies that aim at "always selecting *all* sensors" (regardless of their TTNC) and "only selecting the *minimal* set of sensors" (*i.e.*, those we must calibrate because their TTNC reaches 0). In addition, we also investigate two simple selection strategies. The former consists in selecting all sensors that are co-located with the sensors that form the minimal set ("*local*"). The latter



(a) Movement time vs. number of spots.

(b) Number of calibrators vs. number of spots.

(c) Running time (sec) vs. number of spots.

Figure 3.10 – Scalability of the multi-path planning solvers and proposed heuristic algorithms.

consists in only selecting the sensors that can automatically calibrate with each other without human intervention and that henceforth do not induce additional calibration time ("*local bounded*"). For multi-path planning, we evaluate the performance of two MIP solvers (*GLPK* and *Gurobi*), our two path planning heuristics (NN-based *greedy* and *GA*), and a naïve strategy that sends one calibrator to *each* spot and that should give the highest cost. The algorithm running time is evaluated on the OpenLab cluster of Dept. Computer Science at UCI, where each computing node has 2x Quad-core Intel Xeon 3.0GHz CPU E5450 CPUs.

Indoor vs. Outdoor Results:

In an indoor environment (Figure 3.7), our algorithm (TTNI-driven sensor selection and GA-based multi-path planning) always results in a lower average cost for N ranging from 5 to 200. Compared to the naïve sensor selection strategies, such as "selecting **all** sensors" and "selecting the **minimal** set of sensors" (still considering GA-based multi-path planning), our algorithm combination provides up-to 30% improvement in the long-term average cost. Note that even though our algorithm does not always end up with the lowest cost per iteration (Figure 3.7b), it makes a fair trade-off between the cost and the time (between iterations). Figure 3.7c shows that the time spent to select sensors is short – less than 1 sec for a reasonably complex building incorporating 200 nodes and 60 spots. The same trend also applies in the outdoor environment (Figure 3.8), where the distances between spots are significantly longer (Table 3.1). As the spatial span of the setup grows, the difference among the algorithms becomes more dramatic (note the different y-scale in Figures 3.7a,3.7b and 3.9a,3.9b). Certain naïve approaches are very sensitive to this change (Figure 3.9b, "minimal" and "local bounded"), while our algorithm shows stable performance in both settings.

Normal vs. Emergency Condition Results:

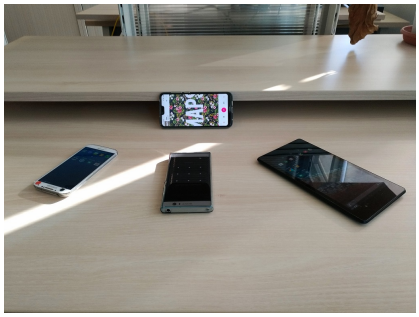
Having demonstrated the effectiveness of our algorithm in indoor and outdoor settings, we further study the performance of our approach in an emergency scenario where the calibration requirements of certain sensor types are increased (Figures 3.10a and 3.10c, note the difference in y-scale). When calibration is required more frequently for some sensors, the naïve/simple approaches suffer from a big increase in average cost, especially when the sensors are deployed densely ($\sum \mathbf{Q}/(N \cdot K) > 0.5$), while the performance of our algorithm and "local bounded" are less affected. We also notice that unlike the simpler approaches ("local bounded" or "minimal"), our TTNI-driven sensor selection algorithm avoids the desynchronisation of the periodical calibrations, while the "local bounded" strategy does so with a small number of sensors (Figure 3.10b).

Scalability Results for the Multi-Path Planning

Figure 3.10 compares the performance of multi-path planning algorithms for the number of spots $L \leqslant 12$: GA provides close-to-optimal solutions but takes 8–10 sec for L=12 (20–60 sec for L=60); the greedy heuristic is much faster (approx. 0.5 sec for L=60), which makes it suitable as an estimator during the sensor selection planning. GLPK and Gurobi (*i.e.*, the MIP solvers) could not terminate within 48 hours for $L \geqslant 15$ so we aborted.

3.4.2 Multi-Hop Multi-Party calibration

We evaluate the performance of the proposed multi-hop, multi-party calibration using the Android app implementation. We specifically run the application on the following Android devices: 1 Samsung Galaxy S5, 2 Asus Google Nexus 7, 1 Crosscall Trekker-X3, and 1 Huawei P9 Lite. So as to assess, the performance/quality of the multi-party calibration, we compare the noise measurements using the mobile phones, with the ones of a professional SLM -*Sound Level Meter* (SVAN 971 from SVANTEK) that meets the IEC61672 international standard. This SLM takes 10 samples per second; each sample corresponds to an average of the sound level power that is sensed during $100ms$, on which an A-weighting is applied. Although our calibration app enables adapting the duration of the calibration and the sampling rate, we use the setting of the SLM in the following evaluation: 10 samples are registered per second and the calibration is performed simultaneously with the sound level meter. Figure 3.11 illustrates some of the undertaken evaluation experiments that we detail hereinafter.



(a) *Indoor calibration:* Three phones perform a multi-party calibration, while another one plays music.

(b) *Underground calibration:* Three persons speak in a quiet parking. All are holding their devices next to a wall, while the person in the middle additionally holds a SLM that serves as ground truth.

(c) *Outdoor calibration:* Three people meet at a bus station, in a bi-directional street that is relatively noisy.

Figure 3.11 – Experiments.

Our evaluation assesses the quality of the regression, and hence of the calibration, accordingly to the following parameters:

- The *variance of the regressand*, $Var(Y)$ – Measures the dispersion of the regression.
- The *coefficient of determination* [42], R^2 – Quantifies the goodness of fit of the regression plane, *i.e.*, the amount of variability in the data set that is explained by the regression. The closer R^2 is to 1 (resp. 0), the greater (resp. lower) the degree of association between Y and the X.

- The *adjusted* R^2 – Evaluates the goodness of fit and penalises the use of additional variables. The adjusted R^2 increases only if new variables improve the regression more than it would be expected by chance. It remains less or equal to R^2 but it can also be negative.
- The *standard deviation of the residual noise*, ε_i.
- The *difference/bias* between the measurements provided by a calibrated device and the ones of the SLM.

We use *box plots* to depict the above parameters when reporting the evaluation results (*e.g.*, see Figure 3.12): the *triangle* denotes the mean, the *line* inside the box indicates the median, the *lines on edge* reflect the range of values, and the *rectangle* corresponds to the Interquartile Range (IQR).

In order to evaluate the multi-hop, multi-party calibration, we consider the following scenarios:

1. Comparison between the simple and robust regression (§ 3.4.2).
2. Comparison between univariate (pairwise) and multivariate (multi-party) regression (§ 3.4.2).
3. Comparison of multi-hop, pairwise with multi-hop, multi-party calibration (§ 3.4.2).
4. Evaluating the impact of the environment (in-door vs out-door) on multi-party calibration (§ 3.4.2).

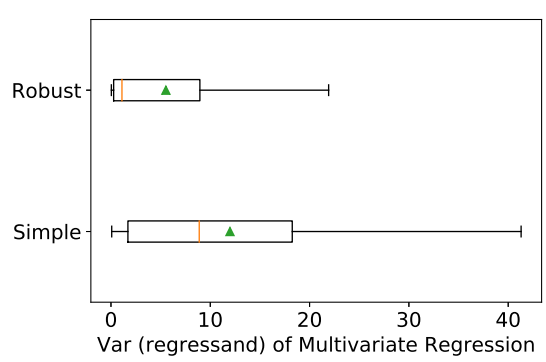
The first three scenarios involve in-door calibration within a room (see Figure 3.11a), while the fourth also considers the underground and out-door cases (see Figures 3.11b and 3.11c). In most of the experiments, the devices were less than 1 meter away from each other, except when we consider the impact of the following sensing conditions: distance, obstacle and in-/out-pocket.

Simple versus Robust Regression

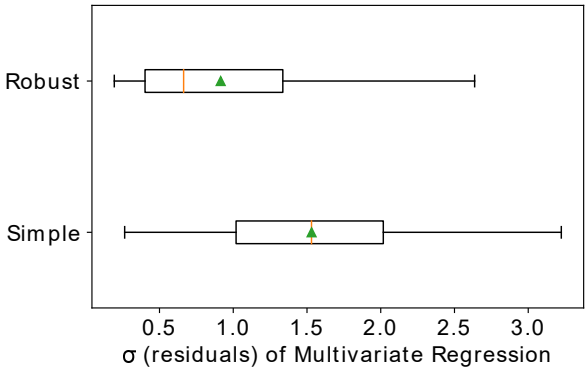
We compare the simple *vs* robust regression using an uncalibrated device (the Crosscall Trekker-X3) and two devices (the 2 Asus Google Nexus 7) that were previously calibrated with the SLM, and that exchange the noise level they collected during $10s$. The experiments were run in-door, 100 times.

Figure 3.12a (resp. 3.12b) compares the robust regression with the simple regression considering the variance of the regressand (resp. the standard deviation of the regression residuals). As shown, the robust regression provides a lower regressand variance in terms of mean and median. In particular, the average value of the variance corresponds to 11.984 with the simple regression and 5.529 with the robust regression. In fact, the robust regression reduces the influence of outliers, which leads to a lower regressand variance and thereby to a more accurate regression. Also, the regression residuals (Figure 3.12b) produced by a robust regression is characterised by a weaker standard deviation: the average value of this standard deviation is equal to 1.531 dB(A) for the simple regression and 0.915 dB(A) for the robust regression. Meanwhile, the IQR of the standard deviation exhibited by the robust regression, *i.e.*, $[0.402dB(A), 1.336dB(A)]$, is much lower, compared to the one provided by the simple regression, *i.e.*, $[1.019dB(A), 2.017dB(A)]$.

In order to evaluate the computational overhead and accuracy of the robust regression, we use the same three devices and we run series of calibrations characterised by various sampling durations; we execute 20 times each of these series.

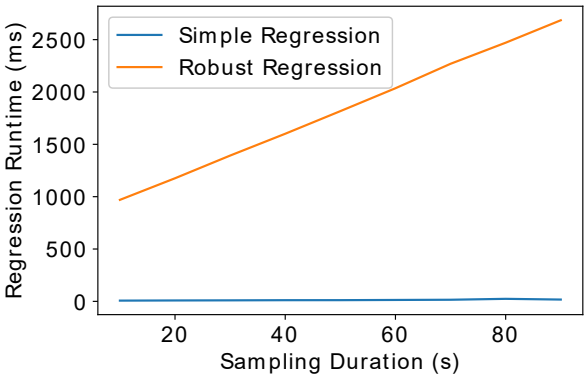


(a) The regressand variance for simple *vs* robust multivariate regressions.

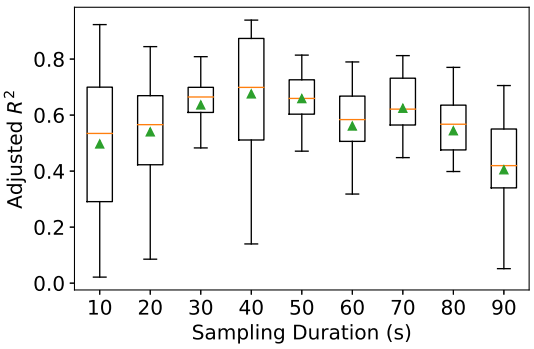


(b) Standard deviation of the residual for simple *vs* robust multivariate regressions.

Figure 3.12 – Simple versus robust multivariate regression



(a) Duration induced by the establishment of a multivariate regression according to the sampling duration.



(b) Adjusted R^2 attained from a robust multivariate regression according to the sampling duration.

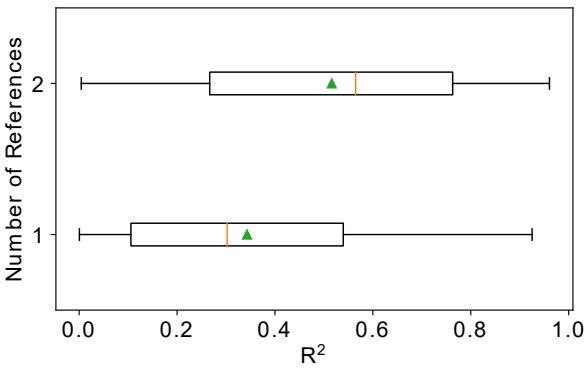
Figure 3.13 – Impact of the sampling duration on the performance of multivariate regression.

The robust regression induces a significant computational overhead (see Figure 3.13a) because many regressions are established and further compared so as to ultimately select the regression that best fits. As an illustration, the simple (resp. robust) regression takes around 7.159 ms (resp. 969.561 ms) when taking 100 sound samples; with 600 samples, the robust regression lasts for around 2035.791 milliseconds. However, this is to be balanced with the resulting adjusted R^2 of the robust regression (Figure 3.13b), which increases and reaches a maximum for a recording duration of 40s and then decreases. This shows that there is no need for unnecessarily increasing unnecessarily the calibration duration.

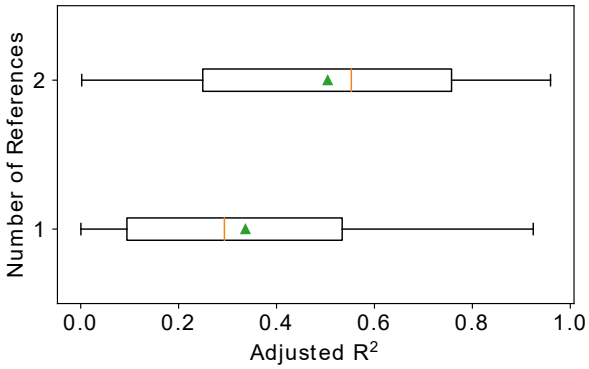
In what follows, we consider only the robust regression.

Univariate versus Multivariate Regression

We used the same three devices as above to conduct one-to-one and one-to-two regressions, 100 times each. Figure 3.14a (resp. 3.14b) evaluates and compares the accuracy of the (robust) univariate vs multivariate regression in terms of R^2 (resp. adjusted R^2). The multivariate regression always has higher R^2 and adjusted R^2 compared to the univariate case. As an illustration, an univariate regression (resp. bivariate regression) generates an average and adjusted



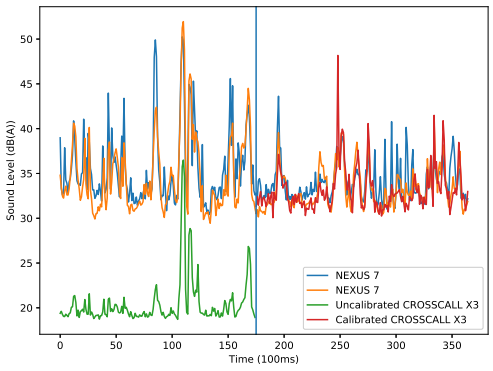
(a) R^2 with univariate *vs* multivariate regression.



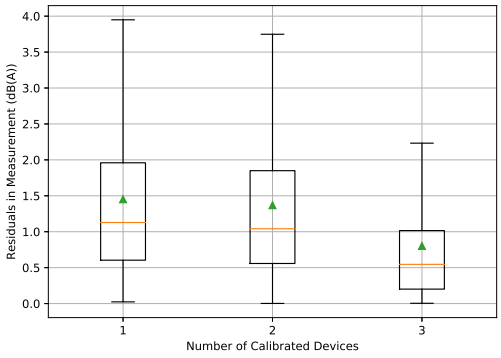
(b) Adjusted R^2 with univariate *vs* multivariate regression.

Figure 3.14 – Univariate versus robust multivariate regression.

R^2 of 0.335 –0.343 as raw– (resp. 0.504 –0.516 as raw). We observe the same trend with the interquartile range. When the number of calibrated devices increases, both the R^2 and adjusted R^2 also increase. This suggests that the multivariate regression provides a better fitting than the –state of the art– univariate approach.



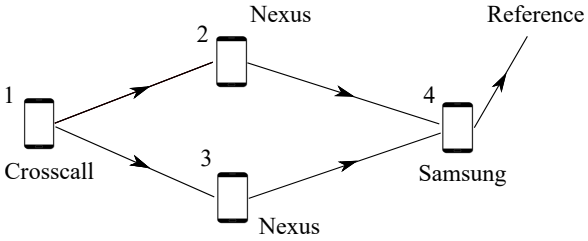
(a) Crosscall Trekker-X3 calibrates using two Asus Nexus 7; the plot splits into two phases: before and after calibration.



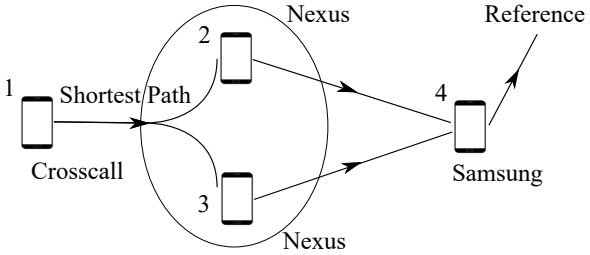
(b) Bias between the calibrated device and the SLM.

Figure 3.15 – Pairwise *vs* multi-party calibration.

We now compare the pairwise (univariate regression) with the multi-party calibration (multivariate regression), using $40s$, instead of $10s$, measurements. We further confronted the noise measurements provided by the newly calibrated smartphone with the ones provided by the SLM over a duration of 100 seconds. Figure 3.15a shows that after a multi-party calibration, the bias between the measurements provided by the device that gets calibrated and the two calibrated devices is drastically reduced. And, quite noteworthy, Figure 3.15b shows that the difference between the measurements provided by the newly calibrated device and the SLM decreases as the number of calibrated devices participating to a calibration rendezvous increases. Overall, the multi-party calibration is more accurate.

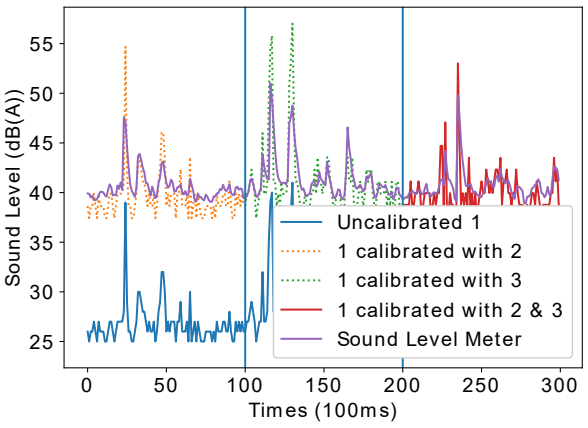


(a) 4 pairwise calibrations lead to the creation of two hyperpaths: $1 \rightarrow 2 \rightarrow 4$ and $1 \rightarrow 3 \rightarrow 4$.

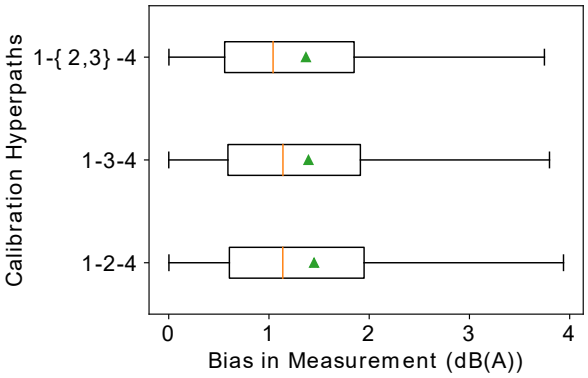


(b) Two pairwise calibrations are followed by a multi-party calibration, resulting in the hyperpath: $1 \rightarrow \{2, 3\} \rightarrow 4$.

Figure 3.16 – Multi-hop, multi-party calibration scenarios.



(a) Three options for the calibration of Device 1: with 2, with 3, or with both 2 & 3.



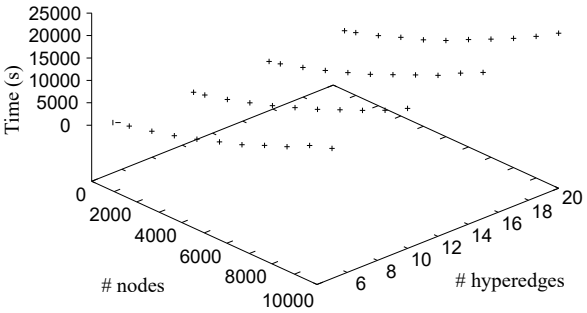
(b) Bias between the device and the sound level meter after calibration via different hyperpaths.

Figure 3.17 – Indoor multi-hop multi-party calibration.

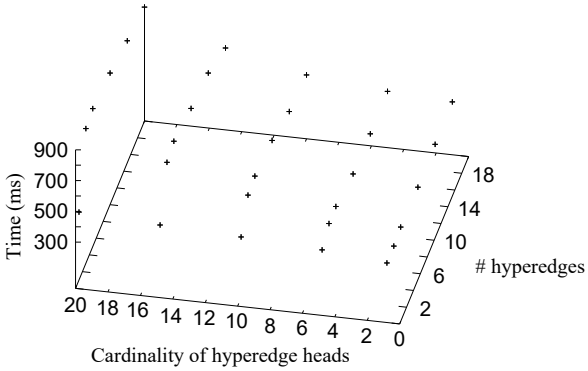
Multi-hop, Multi-Party Calibration

We now investigate a multi-hop, multi-party calibration that takes place during 30s in an indoor environment (see Figure 3.16). In the scenario, Device 4 is the only device that has been calibrated with an end reference node (*i.e.*, the SLM). Device 1 has three options to calibrate: (1) with Device 2 (top Figure 3.16a), (2) with Device 3 (bottom Figure 3.16a), (3) with both Devices 2 and 3 (Figure 3.16b). That is, there are three potential hyperpaths (§ 3.3.2): two correspond to some consecutive pairwise calibrations (*i.e.*, $1 \rightarrow 2 \rightarrow 4 \rightarrow reference$, and $1 \rightarrow 3 \rightarrow 4 \rightarrow reference$) and one refers to the sequence of both multi-party and pairwise calibrations (*i.e.*, $1 \rightarrow \{2, 3\} \rightarrow 4 \rightarrow reference$). Figure 3.17a depicts the measurements of Device 1 before calibration and after calibration for each of the 3 cases; the figure also provides the measurements of the SLM. Each of these hyperpaths (calibrations) is characterised by a specific accuracy, which can be assessed by comparing the measurements provided by the SLM with the ones from Device 1. Figure 3.17b shows that the bias between the measurements provided by Device 1 and the SLM decreases from $1 \rightarrow 2 \rightarrow 4 \rightarrow reference$ and $1 \rightarrow 3 \rightarrow 4 \rightarrow reference$ to $1 \rightarrow \{2, 3\} \rightarrow 4 \rightarrow reference$. The difference between the two former hyperpaths is not significant while the third hyperpath is characterised by a lower bias to the SLM. Table 3.2 considers additional metrics for weighting the hyperedges (*i.e.*, the mean residual R^2 and the residuals σ). In such a case, the shortest hyperpath is the one having either the highest R^2, or the lowest mean residual and the lowest σ (residuals).

In order to evaluate the execution time of the multi-hop, multi-party calibration, we generate a large-scale and random hypergraph. In this hypergraph, each hyperedge is characterised by a head whose cardinality is chosen randomly. In addition, the hypergraph is created so that there exists an hyperpath between the source of the hypergraph and the consolidated node. Figures 3.18a and 3.18b analyse the time needed to compute the shortest hyperpath from the source to the consolidated node. In particular, the time taken to compute the shortest hyperpath (Figure 3.18b) increases linearly with the number of hyperpaths because the length of the hyperpath increases proportionally. Meanwhile, the time taken to compute the shortest path is proportional to the size of the multi-party calibration group.



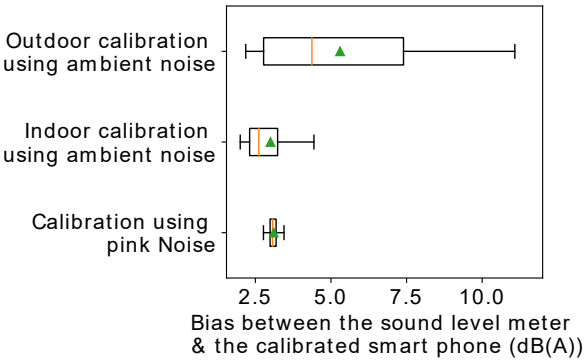
(a) Time induced by computing the shortest hyperpath between the source and the consolidated node with an hypergraph including 500 nodes and 3 end reference nodes.



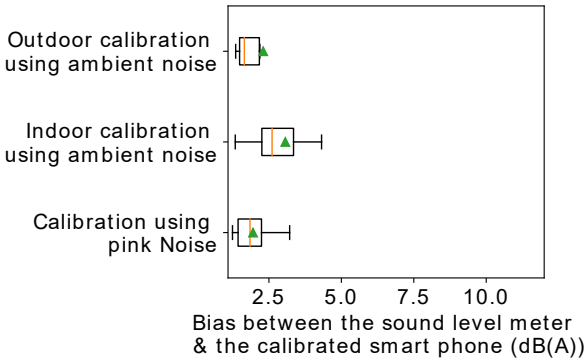
(b) Time induced by computing the shortest hyperpath between the source and the consolidated node with an hypergraph with 3 end reference nodes and meetings involving at most 3 nodes.

Figure 3.18 – Scalability analysis.

Impact of the Sensing Conditions



(a) Bias observed in an indoor environment.

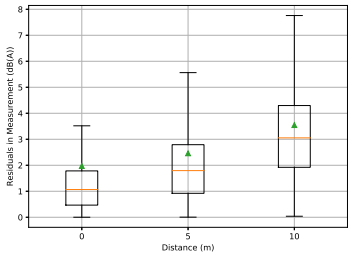


(b) Bias observed in a street characterised by a high traffic.

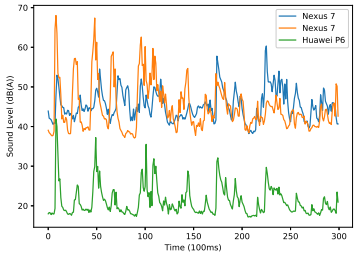
Figure 3.19 – Controlled calibration using pink noise *vs* blind calibration using ambient noise.

We conclude the evaluation with the study of the impact of the sensing environment on the multi-party calibration. We first compare the calibration that is performed in a laboratory using pink noise with the one performed in an indoor/outdoor environment using ambient noise. We consider the calibration accuracy defined by the bias observed between the sound levels provided

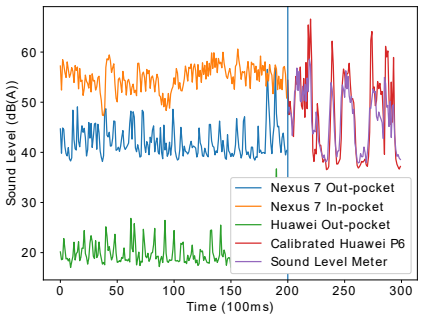
by a SLM and the one given by a newly calibrated device. Figure 3.19a (resp. 3.19b) provides the accuracy in the indoor (resp. outdoor) case. We note that when the calibration is performed in the same environment, the blind and lab calibrations have almost the same accuracy. When the calibrations are performed in different environments, the difference between a blind and a lab calibration remains under acceptable limits (*i.e.*, $\leq 1dB(A)$).



(a) Impact of the distance: The uncalibrated smartphone is placed near a person that speaks and in the middle of the two calibrated smartphones.



(b) Impact of obstacles: The uncalibrated smartphone is placed at a wall corner and 2 calibrated smartphones are placed on the two corner sides, near two persons speaking.



(c) Impact of a smartphone placed in a pocket: The uncalibrated smartphone calibrates with two calibrated smartphones (one in the pocket and the other out-of pocket).

Figure 3.20 – Impact of the calibration conditions.

Finally, we investigate the impact that different operational conditions may have on the calibration, while performing the study in a quiet indoor environment. We specifically consider the impact of: the distance between the devices involved in the calibration rendezvous, the presence of an obstacle, and the presence of a device in a pocket. As expected, the calibration is more accurate (Figure 3.20a) when the devices are very close to each other because they are exposed to the same sound. For instance, the mean residuals after calibration are 1.973, 2.460 and 3.546 dBA for 0, 5 and 10-meter distances, respectively. In general, such a trend is not always observed given that the audio signal can reverberate on obstacles or can be obstructed by obstacles (Figure 3.20b). Even though, the multi-party calibration plays an essential role to mitigate the impact of the obstacle, the residual noise can be used to determine whether the calibration should be effective or abandoned. We end this evaluation by looking at the impact that a smartphone located in a pocket may have (Figure 3.20c). The uncalibrated device favors the smartphone outside the pocket by giving it a high β value compared to the one given to the device in the pocket, therefore showing the robustness of the multi-party calibration. This is thanks to the fact that measurements in different context have low correlation.

3.5 Conclusion

We are witnessing the emergence of some infrastructure-based and crowd-driven applications opening up previously unprecedented ways of linking the human with the physical world. Our increased interconnection with the physical world also holds the promise of an enhanced environmental and social sustainability built upon a better understanding about, and act upon, the physical environment. While expectations are high, monitoring a phenomena at an unprecedented scale is challenging, especially considering that applications must provide the expected meaningful data in any circumstances, regardless of *e.g.*, disturbances, harsh conditions. We are

addressing this challenge through the calibration of the IoT nodes, from the fixed Things that make up the IoT infrastructure to the mobile Things that people carry. Cost-effectively planning the on-site calibration of IoT infrastructure helps in the sustainable long term operation of deployment and in the enhancement of the quality for the gathered observations. We frame multi-sensor calibration planning as an optimisation problem where we propose a two-phase iterative local optimization approach to determine (a) how many calibration iterations are necessary, (b) which sensors should be calibrated at each of these iterations, and (c) the number of mobile calibrators that are required (as well as their respective calibration paths), such that the average cost of all iterations is minimised and under the constraint that the calibrators should not be overloaded. The proposed two-phase iterative local optimisation approach first creates a selection of sensors for each iteration, and introduces new methods (mTSP variants, heuristics) to compute a set of paths for the calibrators based on the selection. Our evaluation shows that the proposed algorithms solve the calibration planning problem effectively compared to naïve/simple solutions. Such calibration solution constitutes a pragmatic approach to support the full-fledged deployment and flexible maintenance of an IoT network infrastructure that is not always uniformly and continuously available. In order to accommodate budget constraints without sacrificing the coverage of the phenomenon of interest, we aim at exploiting all available sensing modalities –fixed as much as mobile– for increased sensing coverage.

Mobile crowdsensing is becoming one of the most promising paradigm supporting social sensing, as it allows to significantly decrease the infrastructure costs and to supply fine-grained information about the phenomenon of interest. In counterpart, the prevalence of low-cost sensors and the presence of human in the sensing loop brings into the picture new research challenges, most of which are yet to be understood and addressed. smartphone sensors are not typically of the same fidelity as task-specific sensors (*e.g.*, a noise pollution monitor vs. smartphone microphone). As a matter of fact, the control of the sensing process is delegated to people that lack of expertise and experience. Inadvertent or erroneous handling of the phone (*e.g.*, phone acting as a noise sensor located in the pocket) may adversely influence the quality of the data, irrespective of the willingness of the participants.

This leads us to introduce a novel macro-calibration problem where numerous devices need to calibrate without involving the end-users. To tackle this new problem, we introduce an opportunistic macro calibration system that leverages the high density of smartphones in the urban environment to support a seamless and multi-party calibration. Our multi-party calibration system coordinates multiple surrounding smartphones and leverages multivariate and robust linear regression to handle outliers. We have implemented and experimented our solution. Our prototyped system contributes to enhancing the accuracy of mobile phone sensing. In particular, our evaluation shows that a multi-party regression is characterised by an accuracy that is always higher, and, in the worst case, equal to the accuracy of a pairwise regression. During our experiments, we investigated the actual impact of faulty measurements that potentially lead to incorrect calibration, *i.e.*, outliers and non relevant measurements obtained by some phone operating within another sensing context (such as those provided by a calibrated phone in a pocket). As presented, our approach leverages robust regression, which systematically discards outliers. In addition, our evaluation shows that our approach allows distinguishing the actual sensing contexts (*e.g.*, measurements taken with the phone(s) in the pocket), which explains by the lack of correlation between the related measurements. Consequently, the phone in the pocket is assigned a negligible weight or is even ignored. Even though the impact of these recognisable faulty measurements remains insignificant, the fact remains that some incorrect calibration parameters or measurements that have been (intentionally or unintentionally) altered may impair a multi-hop calibration. In practice, the regression minimises the squared

distance between the measurements of the uncalibrated and calibrated nodes, which compensates for small discrepancies among the (miscalibrated and well-calibrated) measurements. In general, we argue that the multi-party calibration allows for a more precise characterization of the relationship between crowdsensors than would a pairwise calibration.

Once calibrated – static as much as mobile – sensors provide some observations that are expressed with regards to a common reference system. Their interpretation of the observation depends on the conditions under which they are made. For example, if we are interested in a temperature sensor, knowing whether the sun is hitting is crucial. Same applies with noise observation, previous analysis [83, 165] of the noise observations collected from the opportunistic crowdsensing application Ambiciti (formerly SoundCity) showed that less than 5% of the collected data were of sufficient quality to be considered for use in the mapping of urban noise pollution.

function solveMPPGreedy ($\mathbf{G}$, $\hat{c}$, H, $\boldsymbol{\beta}$) ;
Input : $\mathbf{G}$ – Map; $\hat{c}$ – Maximum workload;
H – Set of selected spots;
$\boldsymbol{\beta}$ – Vector of calibration time at selected spots.
Output: $\{\mathbf{W}\}$ – Set of paths.
Initialise pathSet $\leftarrow$ [ ] ;
while H **is not empty do**
  minInc $\leftarrow +\infty$; minSp $\leftarrow$ minPath $\leftarrow$ **null** ;
  for each *path* **in** *pathSet ; last* $\leftarrow$ *path*$[-1]$ **do**
    oldTime $\leftarrow$ getMoveTime (path) + $\sum_{l \in \text{path}} \beta_l$;
    for each *sp* **in** H **do**
      dCw $\leftarrow G[\text{last}, \text{sp}] + G[\text{sp}, 1] - G[\text{last}, 1]$;
      if *oldTime* + *dCw* $+\beta_{sp} \leqslant \hat{c}$ **then**
        if *dCw* $<$ *minInc* **then**
          minInc $\leftarrow$ dCw ;
          minSp $\leftarrow$ sp ; minPath $\leftarrow$ path ;
        end
      end
    end
  end
  if *minInc* **is finite then** minPath.append (minSp) ;
  else
    for each *sp* **in** H **do**
      if *(dCw* $\leftarrow G[1, sp] + G[sp, 1]) <$ *minInc* **then**
        minInc $\leftarrow$ dCw ; minSp $\leftarrow$ sp ;
      end
    end
    newPath $\leftarrow$ [minSp] ; pathSet.add (newPath) ;
  end
  H.del (minSp) ;
end
return $\{\mathbf{W}\} \leftarrow$ convertPathVecToMatrix (pathSet) ;

Algorithm 1: Nearest-neighbor-based greedy algorithm for the multi-path planning problem.

Input: i: source node
r: consolidated reference node
H_i: hyperpath from i
Output: $SH_{i,r}$: shortest hyperpath from i to r
begin
$H_{i,v} \leftarrow \emptyset$ for all $v \in V$
$|H_{i,i}| \leftarrow 0$, $|H_{i,v}| \leftarrow \infty$ for all $v \in V \backslash i$
$Q \leftarrow V$
while $Q \neq \emptyset$ **do**
$u \leftarrow u'$ subject to $\min_{u' \in Q} |H_{i,u'}|$
forall $E_j \in E$ *so that* $u = tail(E_j)$ **do**
if $|head(E_j)| = 1$ **then**
$v \leftarrow head(E_j)$
if $|H_{i,u}| + w(E_j) < |H_{i,v}|$ **then**
$|H_{i,v}| \leftarrow |H_{i,u}| + w(E_j)$
$H_{i,v} \leftarrow H_{i,u} \cup (\{v\}, E_j)$
end
end
if $|head(E_j)| > 1$ **then**
forall $v \in head(E_j)$ **do**
$H_{v,r} \leftarrow SHPath(v, \mathbb{E} \backslash E_j \backslash H_{i,u}, r)$
end
if *(*$H_{v,r} \neq \emptyset$ *// *$v = r$*) for all* $v \in head(E_j)$ **then**
if $|H_{i,u}| + w(E_j) + \sum_{v \in head(E_j)} |H_{v,r}| < |H_{i,r}|$ **then**
$|H_{i,r}| \leftarrow |H_{i,u}| + w(E_j) + \sum_{v \in head(E_j)} |H_{v,r}|$
$SH_{i,r} \leftarrow H_{i,u} \bigcup E_j \bigcup_{v \in head(E_j)} H_{v,r}$
end
end
end
end
end
if $|H_{i,r}| < |SH_{i,r}|$ **then**
$SH_{i,r} \leftarrow H_{i,r}$
end
return $SH_{i,r}$
end

Algorithm 2: SHPath(i, H, r) - Shortest Hyperpath from i to r in hypergraph H

Input Data		Indoor		Outdoor
		Normal	**Emergency**	
Map	L Num. of Spots	60		63
	G Pairwise Dist.[a]	$\leqslant$53 sec		$\leqslant$73 min
Sensor	K Num. of Types	10		8
	τ Calib. Time	1–30 min		0.25–1 min
	T Calib. Period	14–91 d	7–91 d	28–123 d
Nodes	N Num. of Nodes	100 (varies if independent variable)		
	Sensor Presence	100 (varies if independent variable)		
User Req.	$\hat{c}$ Max. Workload	2 hours		4 hours
	$\boldsymbol{\mu}$ Coefficients	C_{it}=10000, μ_0=1 μ_w=5, μ_c=1		
	T Maintenance P.	360 days		

Table 3.1 – Experimental setup for the performance evaluation.

Weight/Edges	(1, 2)	(1, 3)	(1, {2,3})	(2, 4)	(3, 4)
Adjusted R^2	0.564	0.529	0.570	0.511	0.444
Mean of the residuals	2.504	2.515	2.478	3.5e-14	1.9e-14
σ(residuals)	1.079	1.073	0.786	2.515	2.645

Table 3.2 – Hyperpath weighting

Chapter 4

Leveraging Crowdensors to support Context-aware and cost-effective Crowdsensing

4.1 Introduction

Crowdsensing is becoming one of the most promising paradigms supporting the vision of social sensing: citizens may contribute valuable spatio-temporal observations using the low-cost, yet powerful, sensors embedded in their smartphones/tablets, with the GPS for positioning and the Internet access for uploading. As opposed to mere physical equipment that only senses the environment and transfers data, crowdsensor should be treated as "*social sensor*".

There is a wide range of users' engagements to sensing tasks [94, 65], which depend on how the smartphones' owners contribute observations [94, 65], which may be either: (i) pro-actively, *aka participatory crowdsensing* [74, 90]; or (ii) passively in the background, *aka opportunistic crowdsensing* [37, 102]. *Participatory crowdsensing* allows covering a specific area while maximising the number of tasks that each participant may achieve [91] through careful path planning. On the other hand, *opportunistic crowdsensing* makes it easy to collect a massive amount of observations across time and space. Indeed, there is no burden put on the end-users. The human presence in the sensing loop brings into the picture new research challenges, most of which are yet to be understood and addressed. With participatory sensing, the massive collection of data often necessitates the participants' long term commitments, which are very hard to achieve in practice, mainly due to the practical problem of "hiring" motivated volunteers. Participants lack the technical skills required to set-up and maintain sensors. Participatory sensing is critiqued as a top-down approach allowing non-experts to participate in scientific progress. In other words, participatory sensing is perceived as being closer to a participatory push [62] as opposed to a "form of science developed and enacted by citizens themselves". On the other hand, the opportunistically-collected observations often cover the urban space and time unevenly, due to the mobility of citizens throughout the city. In addition, previous analysis [83, 165] on the noise observations collected from the opportunistic crowdsensing application Ambiciti (formerly SoundCity), showed that less than 5% of the collected data were of sufficient quality to be considered for use in the mapping of urban noise pollution.

Our approach to support the democratisation of environmental crowdsensing, lies in supporting an opportunistic and collaborative crowdsensing. State-of-the-art opportunistic crowdsensing systems address the aforementioned shortcoming – namely the uneven coverage of the urban space and the low-quality of the collected observation – through the centralised analysis – spanning filtering, aggregation and interpolation– of the contributed observations on cloud/edge infrastructure servers [93, 108, 69, 68, 166]. The implemented centralisation then severely limits the adoption of crowdsensing for environmental monitoring due the resulting resource cost and threat to privacy. Instead, our approach is to allow a fully distributed, collaborative approach

to crowdsensing: Crowdsensors interpolate data, and aggregate their respective contributions to the observations of the phenomenon in a collaborative way, so as to overcome the spatio-temporal sparsity and to limit –or even avoid– the use of a centralised infrastructure server. We support a distributed solution to the problem of collection, interpolation and aggregation on the Move, which builds upon the two following trends to support crowdsensing-based environmental monitoring at scale:

(i) We view crowdsensors as "*social sensors*" that often encounter each other during crowdsensing campaigns or as part of their owners' daily routine. When the crowdsensing is *opportunistic*, participants follow their daily routine without being directly involved in the sensing task [37]. Thus, at the micro level, behavioural signatures (i.e., routines) as well as recurrent meeting patterns reflect the underlying relational dynamics of organisations or communities to which the user is affiliated [97]. *Participatory* crowdsensing campaigns bring groups of people together to meet, exchange information and share their findings [10], with the aim of improving the quality of the data collected for better impact on the ground [18]. Compared to the opportunistic crowdsensing, participants in participatory campaigns meet more often and contribute finer scale measurements but cover smaller areas [169]. In both cases, the end-users are more likely to share their information, especially in short meetings, as the risk of losing their anonymity is lower [88].

(ii) The control of the sensing process is delegated to people, who may choose to deliver (or not) relevant information depending on their context as well as their personal constraints, such as financial budget, time, smartphone battery, available network bandwidth. Other factors may also influence the quality of submitted sensing measurements. In particular, smartphone sensors are not typically of the same fidelity as task-specific sensors (*e.g.*, a noise pollution monitor vs. smartphone microphone). Besides, inadvertent erroneous handling of the phone (*e.g.*, phone acting as a noise sensor located in the pocket) coupled with the lack of experience in performing the sensing task (e.g. blurry photo captured by mistake) may adversely influence the quality of the data, irrespective of the willingness of the participants.

(iii) The collaborative and ubiquitous processing of crowdsensor observations improves the overall efficiency of the system in terms of resource consumption [176], regarding both the infrastructure and contributing devices. It also allows end users to collaborate and share knowledge effortlessly and with little or no cost. In particular, D2D collaboration enhances mobile edge computing by enabling the sharing of heterogeneous computing and communication resources between powerful mobile end-devices [36].

The design rationale of our crowdsensing solution directly derives from the social nature and in particular from the location-, context- and time-dependence of the crowdsensors: co-located crowdsensors contribute related observations [96], and may thus collaborate toward improving the provided contributions and avoiding unnecessary duplication of work. More precisely, the crowdsensing tasks that need to be performed at the edge encompass: environment sensing, location provisioning, data processing, and uploading to the cloud. Then, while the replication of physical sensing within a group potentially allows for the gathering of more accurate knowledge, any of the other tasks only need to be performed by the most cost-effective group member(s). The fact that people tend to group [43, 150] as part of their daily activity, follow a daily/weekly routine [61, 47], and are most of the time still [83] further supports such a edge-based collaborative strategy to opportunistic crowdsensing.

Beyond the user's mobility which is relevant to initiate the collaboration with the nearby crowdsensors, to know whether the smartphone is in-/out-pocket, in-/out-door and under-/on-ground is particularly important because the smartphone/sensing device needs to be in a position that enables – yet does not interfere with – sensing the physical characteristics of the surrounding. The sensing context must distinguish between in-pocket and out-pocket observations because the former lead to a quite significant deviation from the ground truth and are thus not readily usable [130]. The same applies to in-door versus out-door – as much as under-ground and on-ground – measurements since aggregating them together to analyse environmental phenomena obviously leads to unreliable results [106]. The context information allows keeping more observations -and even correcting them- for aggregating environmental knowledge, rather than filtering out drastically the crowdsensed data. We qualify as "*context*" the combination of these criteria. Our aim is to create dynamic context-aware collaborative groups of crowdsensors, which are such that the peers within any given group: (i) Stay together for a long-enough time period so as to prevent constant changes and unnecessary grouping reconfiguration; (ii) Operate within the same physical environment (e.g., in-door vs out-door) and hence sense related physical phenomena; and (iii) Perform the same activity so that they behave alike –e.g., it is preferable that all the crowdsensors that collaborate either move together or are still. Then, upon the creation of a group, the middleware distributes the crowdsensing tasks to the most adequate group members according to the nodes' abilities –e.g., a smartphone located in the pocket cannot adequately sense the surrounding sound level. Overall, our approach fosters opportunistic collection of the observations that puts minimum burden on the end-users, while increasing the accuracy and resource-efficiency of the crowdsensing system.

Still, a major issue is to cope with the absence of observations due to the uneven distribution of the crowdsensors and the resulting spatio-temporal sparsity of crowdsensors in some specific areas. Concretely, our solution also allows a fully distributed, collaborative approach to support the interpolation of data and aggregation of the respective contributions to the observations of the phenomenon, so as to overcome the spatio-temporal sparsity and to limit –or even avoid– the use of a centralised infrastructure server. There are many interpolation methods for inferring spatio-temporal phenomena, and the smartphone is becoming increasingly powerful to perform such advanced tasks. At the same time, P2P wireless ad hoc network technology (e.g., WiFi Direct) enables the discovery of nearby mobile devices and the exchange of data between peers, making the collaboration possible [54]. Some crowdsensing systems already exploit the P2P collaboration of crowdsensors as they meet [170, 46, 177]. However, the collaboration primarily deals with handling the relay of data, while deployed static edge servers are in charge of the distributed data aggregation. Our approach leverages the advantage of the former and overcomes the disadvantage of the latter: it implements an opportunistic data relay and analysis – spanning the interpolation and aggregation – on the move, across the crowdsensors.

4.2 Context-Awareness sustaining a Collaborative Crowdsensing

A major challenge is to cope with the dynamics and heterogeneity of the crowdsensors: the user activity, the resources available on the device, and the position of the sensors/smartphones are all criteria that characterise the crowdsensor's contribution to the upper layer application. We qualify as *context* the combination of these criteria.

The accurate monitoring of the physical environment through crowdsensing requires the contextualisation of the contributed observations because the context impacts the quality of

the quantitative observations that mobile crowdsensing gathers.Characterising the context of crowdsensed observations has been extensively studied regarding the recognition of the user activity and more specifically the mobility condition (whether moving and with which mean) [122, 134]. On the other hand, inferring the sensing context with respect to the physical environment under scrutiny has received less attention, while related solutions focus on a single context element and do not account for the diversity of the contributing devices. Yet, it is essential to infer together the context elements that impact the crowdsensing, as they are all equally important. While considering a variety of eligible features, one should account for the variation in the availability of features across the contributing devices. Our approach addresses these requirement by being self-adaptive to the diversity of users (and devices). As a first step, we start from the characterisation of the crowdsensor context, spanning from the physical environment, user's activity and device capabilities. In particular, we provide a ranking over all the candidate features so as to compose the most significant feature set for each classification. Then, the user-centric classification on the user's device may be performed on any subset of the above set so as to account for the diversity of devices and related embedded sensors. The set may further be customised and freely recomposed, trading off energy efficiency and accuracy, to offer adaptiveness.

The sensing context must distinguish between in-pocket and out-pocket observations because the former leads to a quite significant deviation from the ground truth and are thus not readily usable [43]. The same applies to both in-door versus out-door measurements as well as under-ground versus upper-ground readings since aggregating them together to analyse environmental phenomena obviously leads to unreliable results [83].

Overall, the major outcome of the group-based collaboration is:

- A context inference module for crowdsensor to provide explicit context information including both device attributes and user behaviours (§ 4.2.1).
- A context-based grouping to share position/internet and optimise the crowdsensing tasks, so as to decrease power consumption during registration and task assignments (§ 4.3).
- A context-based data aggregation/pre-processing module for crowdsensing groups in order to decrease the Internet traffic toward the cloud (§ 4.4.1).

4.2.1 Context Characterisation

The *context* of a crowdsensor serves assessing the relevance of its peering with nodes in the D2D communication range for the sake of enhanced efficiency, that is, the ability to improve the gathered knowledge prior to transferring it to the cloud in a way that reduces resource-consumption. In addition, determining how long the current context is going to last is critical to ensure that the benefit of the collaborative crowdsensing at the edge outperforms the overhead due to the group management/configuration. The context characterises the crowdsensor's collaboration profile, which subdivides into:

- **PE** (*Physical Environment*) defines the position of the embedded sensors, which influences the given observations of physical phenomena. It detects if the crowdsensor is: in/out-pocket, in/out-door and under/above-ground.
- **UA** (*User Activity*) refers to the mobility (walking, cycling, driving) or non-mobility (still) of the end-user. In order to support the activity recognition, we rely on the activity recognition module [57] implemented by the operating system.

- **DA** (*Device Attributes*) characterises the ability of the device to contribute to the various crowdsensing tasks. The attributes include the available networking and computing capabilities (*i.e.*, type of Internet access –*e.g.*, WiFi–, (upload) bandwidth, remaining battery, CPU frequency, and memory size), the type of embedded sensors (*e.g.*, {"Temperature", "Light", "Pressure", "Humidity", "Sound level"}) together with the related power consumption and accuracy.

The user activity and devices attributes are obtained from the devices. In the following, our aim is focused on the characterisation of the physical environment and on the prediction of how long the current *UA* and *PE* are going to last. The classification of the sensing context of mobile "*crowdsensors*" has been the focus of various papers [131, 73, 77]. However, the existing work focuses on the inference of a single context element, while it is essential to accurately characterise the sensing context as a whole, that is, to identify whether a contributed observation is sensed with the device: in-pocket/out-pocket, in-door/out-door and under-ground/on-ground. Machine learning is an obvious candidate to systematise such a characterisation. Thus, starting from the sensors available on today's smartphones, we identify the features that best serve classifying each of the three elements of the sensing context. We then analyse the performance of candidate updatable learning algorithms to initialise the three resulting classifiers, taking into account their classification accuracy as well as their run-time and memory efficiency, from which we design the hierarchical inference system. In the mean time, we leverage online machine learning in a way similar to the inference of *PE*, to predict how long current *UA* and *PE* should last.

4.2.2 Feature Selection

In order to select the best feature set for each of the three classifications M_{pocket}, M_{door} and M_{ground}, we use a data set of $20K$ which provides the supporting ground truth, using a Crosscall Trekker-X3 phone. The data set covers all the candidate features in all the scenarios to be classified, *i.e.*, in/out-pocket, in/out-door and under/on-ground ; the amount of labelled data for each class is uniform. Among the data supplied by sensors, the candidate features are the following: proximity, temperature, light density, GPS accuracy, abstract RSSI level, GSM RSSI value, WiFi raw RSSI, light density, pressure, humidity. The most relevant features for each of our three classifications are defined based on their significance, which is defined by higher information gain and gain ratio, Gini and ReliefF while we set the required threshold value to 0.1 for both. The top three features for the M_{pocket} classification (see Table 4.1) are proximity, temperature and light density. M_{door} is relatively more complex (Table 4.2) with six candidate features (GPS accuracy, abstract RSSI level, GSM RSSI value, WiFi raw RSSI, light density, temperature) providing an information gain and a gain ratio above 0.1. Although the under-ground scenario (Table 4.3), which is considered as a sub-case of the in-door scenario, shares five of them with the latter and needs new features, *i.e.*, pressure and humidity. However, device models do not always support the sensing of the pressure and humidity, in which case other features remain decisive.

Feature/Metric	Info. gain	Gain ratio	Gini	χ^2	ReliefF
Proximity	0.931	0.720	0.478	17776.819	0.329
Temperature	0.344	0.172	0.213	273.650	0.097
Light density	0.310	0.155	0.169	3758.434	0.034

Table 4.1 – 3 Features classifying In-pocket/Out-pocket

Feature/Metric	Info. gain	Gain ratio	Gini	χ^2	ReliefF
GPS accuracy	0.974	0.715	0.482	9085.097	0.996
Abstract RSSI level	0.794	0.493	0.416	15416.226	0.293
GSM RSSI value	0.738	0.370	0.384	11211.066	0.157
WiFi raw RSSI	0.320	0.180	0.148	437.694	0.133
Light density	0.255	0.127	0.157	5041.293	0.050
Temperature	0.228	0.114	0.125	50.212	0.070

Table 4.2 – 6 Features classifying In-door/Out-door

Feature/Metric	Info. gain	Gain ratio	Gini	χ^2	ReliefF
Abstract RSSI level	0.547	0.340	0.296	11586.949	0.250
GPS accuracy	0.434	0.318	0.222	4182.467	0.528
Temperature	0.485	0.243	0.255	1905.520	0.255
GSM RSSI value	0.463	0.232	0.262	8031.071	0.143
Pressure	0.376	0.188	0.202	6136.352	0.224
WiFi raw RSSI	0.181	0.170	0.086	5594.195	0.103
Humidity	0.276	0.138	0.142	2475.257	0.161

Table 4.3 – 7 Features classifying Under-ground/On-ground

4.2.3 Classifier Initialisation

We further train the classifiers M_{pocket}, M_{door} and M_{ground} using their respective most significant features. Various algorithms are eligible for the classification problem [120] although fewer are updatable. We have specifically selected six candidate updatable algorithms: *Hoeffding Tree (Very Fast Decision Tree), IBk (Instance Based K-nearest neighbours classifier), KStar (Instance-based Learner), LWL (Locally Weighted Learning), updatable Naive Bayes, and SGD (Stochastic Gradient Descent)* [174].

Table 4.4 compares the selected algorithms according to the same four metrics for our three classifications: M_{pocket}, M_{door} and M_{ground}. Our selection criteria is motivated by the size of the model on the (resource-constrained) mobile device, 10-fold Cross Validation Classification Accuracy (CVCA) that reflects the classification accuracy, *i.e.*, the proportion of correctly classified examples, the Online Learning Run-time (OLSR) and the Inference Run-time (IR).

The result for M_{pocket} in Table 4.4 shows that all the classifiers can provide a similar high *CVCA* of about 99%. However, a significant difference appears among the sterilised sizes: *IBk*, *KStar* and *LWL* are storing training instances inside the learning model, which makes the size of the classifier proportional to the size of the training dataset. Instead, *H.Tree*, *NaiveBayes* and *SGD* require a much lower *Size*. *IBk* and *LWL* have an *OLR* greater than $3ms$, while it is less than $1ms$ for the other four. *IBk*, *KStar* and *LWL* all have much longer *IR* than *H.Tree*, *NaiveBayes* and *SGD*. A better cross validation result is discovered for M_{door}: All the algorithms provide the maximum classification accuracy in cross validation of 100%. However, although the dataset is unchanged compared to M_{pocket}, the serialized size of *IBk*, *KStar* and *LWL* increases due to the number of selected features. Besides, the *OLRs* do not change significantly, and are less than $1ms$ except for *IBk* and *LWL*. *IBk*, *KStar* and *LWL* still have a longer *IR* than the other three algorithms. Result for M_{ground} shows that *LWL* and *NaiveBayes* give lower classification accuracy in cross validation than other algorithms, but still over 95%. The high storage cost remains for *IBk*, *KStar* and *LWL* as they require storing historical data. They further cost a much longer *IR* than the other three algorithms. As for *H.Tree*, *NaiveBayes* and

SGD, both their *OLR* and *IR* remain below $1ms$, with the negligible exception of the *IR* of *NaiveBayes* at 1.223. Overall, *IBk*, *KStar* and *LWL* show the highest space and time costs, and we discard them.

Finally, we select Hoeffding Tree, Naive Bayes and SGD as initial classifiers, as they show a fairly small space and time cost as well as the highest accuracy and lowest space/time costs.

Metric/Model	H.Tree	IBk	KStar	LWL	NaiveBayes	SGD
M_{pocket}						
Size (kB)	16	1158	1157	1158	3	5
CVCA (%)	99.1538	99.469	99.350	99.149	98.999	99.180
OLR (ms)	0.020	3.809	0.081	4.344	0.012	0.123
IR (ms)	0.057	10.545	165.844	91.325	1.635	0.018
M_{door}						
Size (kB)	9	1612	1791	1764	4	6
CVCA (%)	100	100	100	100	100	100
OLR (ms)	0.036	5.150	0.062	5.813	0.011	0.172
IR (ms)	0.071	11.823	364.790	109.644	1.610	0.047
M_{ground}						
Size (kB)	13	1763	1763	1763	4	6
CVCA (%)	100	100	100	98.060	97.105	100
OLR (ms)	0.024	4.628	0.062	6.720	0.009	0.111
IR (ms)	0.061	15.160	238.916	128.149	1.223	0.018

Table 4.4 – Initial learning models.

In order to predict how long the current *UA* and *PE* are going to last, we leverage online machine learning in a way similar to the inference of *PE*: we predict the duration of the current *UA* (resp. *PE*) according to the current time and *UA* (resp. *PE*). The prediction of the *UA/PE* duration is a regression problem, rather than a classification. Many online learning algorithms may address the prediction model but fewer deal with regression. We have investigated three eligible algorithms: IBk (Instance Based k-nearest neighbour algorithm), KStar (an instance-based learner), and LWL (Locally Weighted Learning) [174]. We selected *LWL* as training algorithm because it provides the lowest RMSE and latency.

4.2.4 Online Personalising

The design rationale of our online learning solution is the following: The initially trained classifier is deployed on the participating devices at the time of the installation of the embedding crowdsensing middleware/application. While the inference of the sensing context is running on the device, feedback is requested from the user to assess the correctness of the inference result. The collected opportunistic feedback is then converted to a labelled training instance that updates the current learning model.

Through the online learning, the initial classifier gets personalised on various device models, according to diverse user behaviours and usage scenarios. In particular, the features that are not available locally become non-contributing to the related classifiers as they get updated. There is a well-known risk associated with online learning that is reaching a local optimum but not a global one as the classification accuracy may not be convex due to the continuous update. We address this limitation through opportunistic feedback where the user is given the opportunity to notify wrong context inferences.

Gathering feedback should be limited as much as possible to minimise the burden on the user, while still enhancing the accuracy of our three classifiers M_{pocket}, M_{door} and M_{ground} over time. We achieve the above by applying a hierarchical inference and update of the classifiers. While more detail about the hierarchical inference (along with the related Algorithm) can be found in [51], the design rational of the hierarchical follows from the predominant role of the in-pocket classifier over the two others and of the in-door classifier over the under-ground one when sensing the physical environment. Specifically, a crowdsensed measurement is relevant if out-pocket. Thus, the in-door/out-door detection is meaningful only when the device is out-pocket and ready for sensing. Further, the under-ground/on-ground case is a sub-scenario of the in-door situation. Similarly, requesting the user's feedback is sensible only if the user is in the position to easily do so. Typically, requesting feedback when the device is in the pocket is too cumbersome for the user.

4.3 Assessing the Crowdsensor Utilities

We introduce a set of utility functions to evaluate the extent to which crowdsensors are eligible to carry out the various crowdsensing tasks. The following crowdsensing tasks must be implemented within any collaborative group: *Coordinator* (implementing the D2D network access point for the group and assigning the crowdsensing tasks to the connected crowdsensors, *i.e.,neighbours*), *Location provider* (supplying geographical coordinates), *Internet proxy* (providing Internet access and thus transferring data to the cloud), *Data aggregator* (analysing together the collected data locally before sending to the cloud to, *e.g.*, calibrate the sensors [149], analyse the data [165]), and, of course, *Sensors*. In practice, we rely on the following classical squashing function f for the normalisation of the values used in the computation and comparison of the utilities:

$$f(x, k, x_0) = \frac{1}{1 + e^{k(x_0 - x)}}$$

where: the range of f lays into $(0, 1)$, k is the logistic growth rate or steepness of the curve, and x_0 is the x-value of the midpoint of sigmoid, while x is the variable to be normalised. The values of k and x_0 are set according to the domain of the specific x. In the following, we denote k_f and x_f the values of k and x_0 for a given function f, while the actual values of the various parameters have been set during the prototype evaluation. Next, we further define the utility functions for each crowdsensing tasks, relying on the squashing function that normalises the values (as defined in Table 4.5)

Coordinator

- The selection of the coordinator is based on the following criteria:

- *The number of neighbours* N_i has a significant impact on the overall performance because increasing the number of collaborators enables analysing more data locally and reducing the transfer of data to the cloud. In contrast, there is no benefit in creating groups with too few crowdsensors for which the minimum value δ depends on the application.

- *The occurrences of collaboration* h_i between a crowdsensor i and its neighbours is another relevant parameter as it reveals the relationship between crowdsensors materialised by the sharing of daily routines, habits, activities.

- *The UA and PE (except pocket)* d_i of the crowdsensor should last sufficiently long.

Parameter	Function	Parameter(s)
Sufficient number of neighbours	$\Delta_i = f(\max\{0, \lvert N_i\rvert - \delta\}, k_\Delta, x_\Delta)$	N_i : neighbours δ: minimum value
Collaboration occurences	$h_i = f(\frac{\sum_{j \in N_i} t_{collab}(i,j)}{\lvert N_i\rvert}, k_h, x_h)$	$t_{collab}(i,j)$: number of times i collaborated with j
Context (*i.e.*, UA, PE) durability	$d_i = f(\min\{dur(UA), dur(PE)\}, k_d, x_d)$	$dur(UA)$, resp. $dur(PE)$ predicted duration of the current UA, resp. PE
Remaining battery capacity	$b_i = f(bat_i, k_b, x_b)$	bat_i: remaining battery
Accuracy versus power	$l_i = f(acc_l, k_l, x_l) - f(pow_l, k_l, x_l)$	acc_l: accuracy pow: power
Upstream bandwidth versus power	$n_i = f(bw_{up}, k_n, x_n) - f(pow_{net}, k_{np}, x_{np})$	bw_{up}: Bandwidth pow_{net}: power consumed for transmission
Accuracy vs power	$d'_i = f(dur(PE_{out}), k_d, x_d)$	$dur(PE_{out})$: duration of the crowdsensor being out of the pocket

Table 4.5 – Set of parameters used to measure the utility associated with a task

- The *remaining battery capacity* b_i of the crowdsensor should obviously be taken into account.

Next, provided the weights w_Δ, w_h, w_d, w_b, all $\in [0, 1]$, set for the above criteria, we define the utility $u_c(i)$ of the crowdsensor i associated with acting as coordinator as the weighted sum of the above functions:

$$u_c(i) = w_\Delta.\Delta_i + w_h.h_i + w_d.d_i + w_b.b_i$$

Location provider

On a smartphone, the location is either GPS- or network-based. While the GPS location brings higher accuracy out-door, it also comes with higher energy consumption and latency, compared to the network-based location. The utility function for crowdsensor i of being a location provider is thus defined as $u_l(i)$, which accounts for the location service source[1], location accuracy and remaining battery capacity:

$$u_l(i) = l_i + w_d.d_i + w_b.b_i$$

Internet proxy

The Internet proxy transfers the (analysed) data provided by the group of crowdsensors to the cloud server. The service may be provided using either long range cellular or short range WiFi transmission, while we assume that a node supporting both networks will offer the WiFi based transmission by default. The utility function $u_p(i))$ for crowdsensor i acting as Internet proxy then accounts for the Internet connection interface, up-link network bandwidth, and remaining battery capacity:

$$u_p(i) = n_i + w_d.d_i + w_b.b_i$$

1. Note that when the location service not operational, $l_i = -\infty$.

Data aggregator

The data aggregator provides in charge the analysis of the sensing data collected locally. While lightweight data processing can be performed by the coordinator or by the proxy, complex data analysis may be delegated to a dedicated device that holds the necessary processing capabilities (spanning available memory, CPU frequency and remaining battery capacity). This results in the following definition for the supporting utility function $u_a(i)$:

$$u_a(i) \quad = \quad w_c.[f(cpu_i, k_c, x_c) \; - \; f(pow_c, k_{cp}, x_{cp})] \; + \; w_r.f(mem_i, k_r, x_r) \; + \; w_d.d_i \; + \; w_b.b_i$$

where: cpu_i is the CPU frequency, pow_c is the power consumption of the CPU, mem_i is the available memory, and coefficients w_c, w_r are the weights set for the metrics.

Sensors

The utility of a crowdsensor to carry out the sensing task depends on the accuracy of the contributed observations and their power consumption. In particular, a crowdsensor within a pocket/bag is ignored, which we filter out using the duration of the crowdsensor being out of the pocket (as defined by the PE_{out} value of PE):
which leads to the following utility function, $u_s(i) \in (-1, 1 + w_d)$:

$$u_s(i) = d'_i.[f(acc_i^s, k_a, x_a) - f(pow_i^s, k_{sp}, x_{sp}) + w_d.d_i]$$

where: acc_i^s (resp. pow_i^s) is the accuracy (resp. power consumption) of the sensor of type s on crowdsensor i.

Utility-based Grouping

Provided the utility functions, the grouping algorithm leverages the communication and discovery protocols implemented at the link layer by state of the art D2D communication technologies. Without loss of generality, our grouping algorithm builds upon the WiFi Direct technology [9]. In a nutshell, WiFi Direct establishes a D2D opportunistic network through the discovery of peer nodes followed by the election of the node acting as the network's access point according to the criteria provided by the upper layer (*i.e.*, in our case, the coordinator utility value provided by the middleware). Furthermore, the grouping algorithm runs on every node i, either on-demand or on a periodic basis according to the network's dynamics. Once a group is created, the arrival/departure of nodes is detected at the link layer by the underlying D2D protocol. This enables two approaches to the reconfiguration of the group: (i) on a periodic basis, or (ii) on-demand upon the detection of the departure/arrival of a group member. The latter performs very well and induces almost no cost when very few topology changes occur and when the context evolves slowly. On the other hand, it may lead to a unnecessarily high traffic and constant reassignment in a highly dynamic environment. Keeping in mind that, in practice, users are still most of the time (and hence evolving in the same context), by default we adopt an on-demand approach (based on context change). Nevertheless, the middleware switches to periodic re-assignment when the context gets highly dynamic. Subsequently, our work focuses on a distributed interpolation and aggregation approach running on the crowdsensors to achieve both higher sensing quality and efficiency.

4.4 Opportunistic Aggregation and Interpolation

A mobile crowdsensing application dealing with urban pollution monitoring (*e.g.*, noise, air quality, water quality, etc.) needs to sense, pre-process, interpolate, then aggregate and relay measurements in a P2P way so as to favour computation at the edge and thereby limit the resource and financial costs associated with the use of the cloud infrastructure. Concretely, we attempt to support a fully distributed, collaborative approach to crowdsensing, in which crowdsensors further interpolate data, and aggregate their respective contributions to the observations of the phenomenon in a collaborative way. The intent is to overcome the spatio-temporal sparsity and to limit –or even avoid– the use of a centralised infrastructure server. There are many interpolation methods for inferring spatio-temporal phenomena ; the smartphone is becoming increasingly powerful to perform such advanced tasks. However, the collaboration [170, 46, 177] primarily deals with handling the relay of data, while deployed static edge servers are in charge of the distributed data aggregation. Our approach leverages the advantage of the former and overcomes the disadvantage of the latter: it implements an opportunistic data relay and analysis on the move, across the crowdsensors.

In particular, our solution supports the repeated monitoring of a physical phenomena over (a possibly large) area over a given time period $\mathcal{D}$ using the contributions of the m crowdsensors. The data that each crowdsensor collects is represented as a concise 3-dimensional tensor where the first two dimensions refer to the spatial space and the third one to the temporal domain. In particular, we discretise the target region into a $I \times J$ area, which contains equally spaced grid points. We also discretise $\mathcal{D}$ into K time slots of equal duration.

We denote $\mathcal{Y}_s \in \mathbb{R}^{I \times J \times K}$ the tensor that crowdsensor s ($1 \leq s \leq m$) maintains and that is such that entry $y(x) \in \mathbb{R}$ at position $x := (i, j, k) \in \mathbb{R}^3$ is the average of the measurements collected by s over the area indexed by (i, j) during the time interval indexed by k. In other words, any crowdsensor s contributes a tensor $\mathcal{Y}_s$ that provides a sparse/incomplete observation of the physical phenomenon according to s behaviour. The middleware then allows the opportunistic combination of the various tensors $\mathcal{Y}_s$ ($1 \leq s \leq m$) using interpolation and aggregation so as to compute an overall $\mathcal{Y}$.

The periodic process runs on every participating crowdsensor to compute $\mathcal{Y}$. The process iterates over time windows of duration $\mathcal{D}$. Within a given time window $T_{+\mathcal{D}}$, the process runs in parallel: (i) the collection of the measurements provided by the embedded sensors to update the tensor $\mathcal{Y}_s(T_{+\mathcal{D}})$ of the current time window, and (ii) the opportunistic aggregation of the local tensor of the previous time window T with the ones of peers as they meet (see detail in § 4.4.2). At the end of the time window, GPR-based spatio-temporal interpolation is applied on the associated local tensor to recover missing values (see detail in § 4.4.1). We highlight that the interpolation is run locally, only once and prior to the aggregation process run over the next time window because: (i) the number of occurrences must be minimised due to the high cost of the interpolation computation, and (ii) the local interpolated tensor allows assessing the quality of the local measurements against the ones of the peers that the node meets, which the aggregation process leverages. Finally, at the end of the current time window $T_{+\mathcal{D}}$, the node ultimately sends its local tensor to the cloud, unless it got aggregated and relayed by another peer.

4.4.1 Spatio-temporal Interpolation

Given a sparse tensor $\mathcal{Y}_s$ resulting from the averaging of the local measurements collected at s over time duration $\mathcal{D}$, interpolation allows completing the tensor by estimating missing

observations. The dense tensor $\hat{\mathcal{Y}}_s$ denotes the resulting (denser) tensor. The quality of the estimation can be established based on the approximation error (*i.e.*, residual) at point x, that is: $e(x) := |y(x) - \hat{y}(x)|$ where we recall that the function $y : \mathbb{R}^3 \mapsto \mathbb{R}$ maps an arbitrary point $x := (i, j, k)$ to its observation value $y(x)$. The overall residual is then given by $\mathcal{E} = \|\mathcal{Y} - \hat{\mathcal{Y}}\|$.

Let Ω be the set of observation points collected by a given crowdsensor. The boolean mask tensor $\mathcal{M} \in \mathbb{B}^{I \times J \times K}$ is defined such that $m(x) = 1$ if there is a corresponding value within Ω, and $m(x) = 0$ otherwise. Thus, $\mathcal{M} * \mathcal{Y}$ provides the actual observations (*i.e.*, ground truth as sensed). When estimating $\hat{\mathcal{Y}}$ based on a sparse tensor $\mathcal{Y}$ with mask $\mathcal{M}$, we seek to minimise the following loss function, which is associated with the approximation $\hat{\mathcal{Y}}$:

$$J(\mathcal{Y}, \hat{\mathcal{Y}}) := \frac{1}{2} \sum_{x \in \Omega} e(x)^2 = \frac{1}{2} \|\mathcal{M} * (\hat{\mathcal{Y}} - \mathcal{Y})\|^2$$

where: $\| \cdot \|$ denotes the Euclidean norm of a tensor, and $*$ represents the element-wise multiplication.

We leverage *Gaussian Process Regression* (GPR) [132] to compute $\hat{\mathcal{Y}}$ out of $\mathcal{Y}$. That is, we assume that y follows a Gaussian Process (Gaussian distribution over functions), *i.e.*,:

$$y(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$$

where: $\mu(x) = \mathbb{E}[y(x)]$ refers to the mean function, and $k(x, x')$ is the covariance matrix, *i.e.*, the kernel of the GPR, which verifies $k(x, x') = \mathbb{E}[(y(x) - \mu(x))(y(x') - \mu(x'))]$.

The covariance function is a crucial ingredient of GPR as it encodes the notion of similarity between two nearby data points x and x' on the basis that observations that are close to each other (in both time and space) are likely to have higher correlation. Various families of covariance functions exist (see [132] for an overview), including, *e.g.*, squared exponential, polynomial, and Matérn class. In our case, the Matérn class [110] induced the lowest error and runtime compared to alternative functions.

We assume that the observation y is noisy, which is more realistic. We therefore consider a regression that aims at establishing $y = y(x) + \epsilon$ where: the function $y(x)$ follows a Gaussian Process, *i.e.*, $y(x) \sim \mathcal{GP}(\mu, k)$, and the noise is additive, independent, and corresponds to an identically distributed Gaussian noise: $\epsilon \sim \mathcal{N}(0, \sigma_e^2)$. Furthermore, given n observed values $Y = [y_1, \cdots, y_n]^\top$ obtained at the location points $X = [x_1, \cdots, x_n]^\top$, the joint distribution of $[y(x_1), \cdots, y(x_n)]^\top$ follows a Gaussian distribution:

$$[y(x_1), \cdots, y(x_n)]^\top \sim \mathcal{N}(\boldsymbol{\mu}, \mathcal{K})$$

where: $\boldsymbol{\mu} = [\mu(x_1), \cdots, \mu(x_n)]^\top$ refers to the mean vector, and $\mathcal{K}$ is a $n \times n$ covariance matrix with $\mathcal{K}_{ij} = k(x_i, x_j)$.

Our interpolation aims at inferring $Y^* = y(X^*)$, which includes m unobserved points stored in the tensor $X^* = [x_1^*, \cdots, x_m^*]^\top$.
The inferred tensor Y^* follows a Gaussian distribution:

$$p(Y^*|X^*, X, Y) = \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2 + \sigma_e^2 \mathcal{I})$$

With a mean and variance of Y^* respectively estimated as:

$$\hat{\boldsymbol{\mu}} = \mathcal{K}(X, X^*)^\top [\mathcal{K}(X, X) + \sigma_e^2 \mathcal{I}]^{-1} Y$$
$$\hat{\boldsymbol{\sigma}^2} = \mathcal{K}(X^*, X^*) - \mathcal{K}(X, X^*)^\top [\mathcal{K}(X, X) + \sigma_e^2 \mathcal{I}]^{-1} \mathcal{K}(X, X^*)$$

Once the GPR model is trained, the mean value $\hat{\mu}$ and variance $\hat{\sigma}$ of any given point x are inferred. In addition, the complete approximation tensor $\hat{\mathcal{Y}}$ is provided along with the variance tensor $\hat{\mathbf{\Sigma}}^{\mathbf{2}} \in \mathbb{R}^{I \times J \times K}$, in which each element $\hat{\sigma}^2$ provides the variance of the corresponding $\hat{\mu}$, namely $\hat{y}(x)$. Note that GPR is computationally demanding as the training scales in $O(n^3)$ with n being the number of observations. Thus, applying such a regression over the overall dataset at the cloud incurs significant computation and financial costs. As an alternative, the middleware distributes the training and inference load over the crowdsensors in an opportunistic way.

4.4.2 Opportunistic Aggregation

The design of the middleware is such that it favours the occurrences of aggregation among the crowdsensors over the ones of the cloud for the sake of lowering the overall computation and communication costs at the cloud, while maintaining the same data quality. To achieve this, our middleware implements an opportunistic aggregation approach. Upon a meeting (involving a shared D2D communication range) of two crowdsensors s and s', the middleware selects one of the two to aggregate their respective tensors $\hat{\mathcal{Y}}_s$ and $\hat{\mathcal{Y}}_{s'}$. Assuming the selected node is s, then $\hat{\mathcal{Y}}_s$ is updated as the aggregation of $\hat{\mathcal{Y}}_s$ and $\hat{\mathcal{Y}}_{s'}$, and $\hat{\mathcal{Y}}_{s'}$ is set to the null tensor. Then, s relays $\hat{\mathcal{Y}}_s$ till it meets another crowdsensor or till the current time window $T_{+\mathcal{D}}$ expires in which case s sends the tensor to the cloud.

The crowdsensor that is the best suited to act as the relay node is the one that will meet the highest number of nodes in the future and thus minimises the occurrences of A_c. We consider that the respective inference quality of the tensors somehow hints on the probability of encountering other crowdsensors in the future. Evaluation further shows that it is a relevant criterion. Precisely, we use the inference quality as defined by the following loss function:

$$D(\hat{\mathcal{Y}}_s, \hat{\mathcal{Y}}_{s'}) := \frac{\|(\mathcal{M}_{s'} * \neg\mathcal{M}_s) * (\hat{\mathcal{Y}}_{s'} - \hat{\mathcal{Y}}_s)\|^2}{2\|\mathcal{M}_{s'} * \neg\mathcal{M}_s\|^2}$$

where $\neg$ corresponds to the NOT Boolean operation and $\mathcal{M}_s$ and $\mathcal{M}_{s'}$ correspond to the mask tensors of s and s'.
The crowdsensor with the lowest loss function performs the actual aggregation. In the following, we describe the aggregation function that crowdsensor s runs provided the tensor $\mathcal{Y}_{s'}$ from s', to compute the new tensor $\mathcal{Y}_{ss'}$. The aggregation function (see Eq. 4.2) sums the following :

- the sensor measurements that are only provided by s omitting the interpolated measurements of s': $\mathcal{Y}_s * (\mathcal{M}_s * \neg\mathcal{M}_{s'})$,
- the sensor measurements provided by s' omitting the ones of s' that are interpolated: $\mathcal{Y}_{s'} * (\mathcal{M}_{s'} * \neg\mathcal{M}_s)$,
- the actual sensor measurements provided by both s and s', which are averaged together: $\frac{n_s\mathcal{Y}_s + n_{s'}\mathcal{Y}_{s'}}{n_s + n_{s'}} * (\mathcal{M}_s * \mathcal{M}_{s'})$ with m (resp. n) measurements denoting the amount of measurements collected by crowdsensor s (resp. s'). Note that the above merged average is built upon a simple algebraic expression of addition that works well to fuse the measurements provided by, *e.g.*, temperature sensor. Such a merged average should be tailored to deal with observations that do not follow an algebraic expression. This is, *e.g.*, the case when merging the average sound level provided by mobile crowdsensors, which is the focus of the experiments.
- The two values resulting from interpolation, which are aggregated using the Generalised Product-of-Expert of GPR, as detailed below.

Overall, the aggregation function performed by crowdsensor s, provided the tensor $\mathcal{Y}_{s'}$ from s', to compute the new tensor $\mathcal{Y}_{ss'}$, is the following:

$$\mathcal{Y}_{ss'} = \mathcal{Y}_s * (\mathcal{M}_s * \neg\mathcal{M}_{s'}) + \mathcal{Y}_{s'} * (\mathcal{M}_{s'} * \neg\mathcal{M}_s) + \frac{n_s\mathcal{Y}_s + n_{s'}\mathcal{Y}_{s'}}{n_s + n_{s'}} * (\mathcal{M}_s * \mathcal{M}_{s'}) \tag{4.1}$$

$$+ (\beta_s\mathcal{Y}_s/\boldsymbol{\Sigma}_s^2 + \beta_{s'}\mathcal{Y}_{s'}/\boldsymbol{\Sigma}_{s'}^2)/(\beta_s\boldsymbol{\Sigma}_s^{-2} + \beta_{s'}\boldsymbol{\Sigma}_{s'}^{-2}) * (\neg\mathcal{M}_s * \neg\mathcal{M}_{s'}) \tag{4.2}$$

Generalised Product-of-Expert is a method that allows combining predicted results that have been inferred by several experts (*e.g.*, sensor measurements and interpolation). In particular, it enables weighting the respective importance of the experts according to the reliability of their prediction. The aggregation of multiple GPR inferences is a generalised product-of-expert, which accounts for multiple inference distributions p_s of an arbitrary point x^*. According to [33], it combines many Gaussian distributions with mean $\hat{\mu}_s(x^*)$ and variance $\hat{\sigma}_s^2(x^*)$ from any crowdsensor s, and it is defined based on:

$$\hat{\mu}(x^*) = \hat{\sigma}^2(x^*)\sum_{s=1}^{m}\beta_s(x^*)\hat{\sigma}_s^{-2}(x^*)\hat{\mu}_s(x^*)$$

$$\hat{\sigma}^2(x^*) = [\sum_{s=1}^{m}\beta_s(x^*)\hat{\sigma}_s^{-2}(x^*)]^{-1}$$

with a weighting parameter β_s that allows tuning the relative importance of a crowdsensor s according to the reliability of its prediction. Regarding the value of β, we distinguish whether the aggregation is performed at the cloud or at a crowdsensor. In the former case, the cloud considers all the incoming data as equivalent; thereby, $\beta_s = \beta_{s'} = 1$. On the other hand, the opportunistic aggregation on the move is asymmetric, as captured by the loss function D. Our evaluation shows that assigning a greater β to the crowdsensor acting as the merge base (*i.e.*, resulting in the least loss) leads to a higher aggregation accuracy.

4.5 Performance Evaluation

4.5.1 User-centric Context Inference

Our primary aim is to evaluate the accuracy of the proposed classifications and the amount of feedback required for personalization. In order to select the most efficient machine learning algorithm for the classifications, we assess candidate algorithms using both negative and positive feedback. The final solution relies on the Hoeffding Tree and uses only negative feedback hierarchically so as to keep the amount of feedback to a minimum and limit the burden put on the end user. The initial classifier is trained once on a computer and then deployed on the smartphones where it evolves.

We evaluate our updatable approach relying on the training dataset $DATASET_1$ (see Section 4.2.2), and a new testing dataset $DATASET_2$.

$DATASET_2$**:** Similarly to $DATASET_1$, $DATASET_2$ contains $20k$ instances, each embedding three labels representing the ground truth, and covers all the relevant scenarios (i.e., in/out-pocket, in/out-door and under/on-ground) uniformly. Differently to $DATASET_1$, the environment sensors, including temperature, humidity, pressure are not available on the contributing device Xiaomi Redmi Note 4, and the available sensors are from a distinct manufacturer. In addition, the user can switch off the WiFi module. Furthermore, the data gathered for $DATASET_1$ and $DATASET_2$ correspond to two different physical environments as they were collected in two different city areas and at different time period (i.e., different months).

The simulation-based experiments reported in the two next sections are run multiple times and break down into the following three steps:

1. The initial classifier, which is generated from batch training using $DATASET_1$ is imported and assessed against the entire $DATASET_2$ to provide an initial classification accuracy for M_{pocket}, M_{door} and M_{ground} respectively.

2. The opportunism is simulated by randomly selecting 30 instances from $DATASET_2$. The 30 instances are classified sequentially using their corresponding feature vector. Then, the feedback is simulated: the inference result is compared to the ground truth label in the instance. If the classification is correct, the instance represents a positive feedback. Otherwise, it represents a negative feedback.

3. The simulated feedback is used to update the classifier. Once a classifier is updated, it is evaluated on the entire $DATASET_2$ again to provide a new classification accuracy. Thus, every feedback iteration has a corresponding classification accuracy.

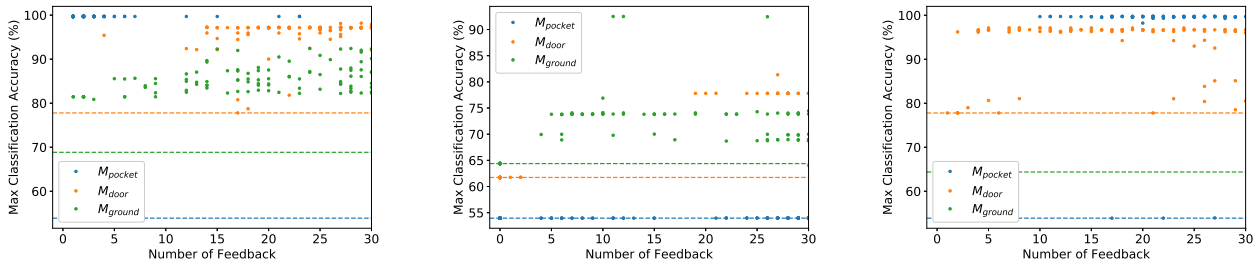
$DATASET_2$ is shuffled in each run. Therefore the opportunistic feedback are randomised. The baseline is the initial classification accuracy performed by the initial classifier on $DATASET_2$ before any update.

Assessing the online learning algorithms and the type of feedback

Since our objective is to update the classifier for a particular user, it is important to (i) investigate the amount of feedback required to achieve a high inference accuracy and (ii) identify the algorithm providing the highest accuracy with the least feedback. For M_{pocket}, M_{door} and M_{ground}, we evaluate the three algorithms Hoeffding Tree, Naive Bayes and SGD, as selected in Section 4.2.3. For each learning algorithm in each classifier M, the λ parameter is configured constantly as 10 to boost the online learning process.

We have run the experiment 100 times and in each run, the volume of feedback reaching the highest accuracy along with the classification accuracy itself are recorded. Note that in this experiment, the inference and update are not hierarchical, which means that the update for each M is independent of each other.

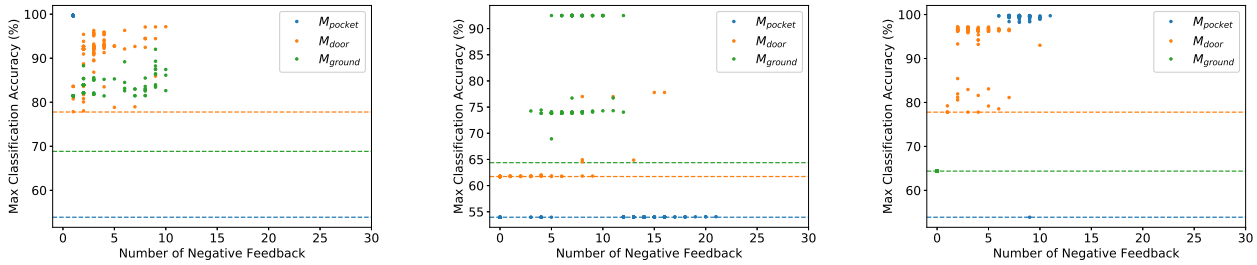
Figure 4.1 shows the classification accuracy for the three algorithms over the three classifications, when both positive and negative feedback are requested from the user. We observe that the algorithms have different performance across the classifications. For M_{pocket}, the accuracy of Naive Bayes does not increase at all with the volume of feedback, while both Hoeffding Tree and SGD can increase their accuracy after few feedback, and Hoeffding Tree requires less feedback to reach the highest accuracy. For M_{door}, Naive Bayes shows both lower initial classification accuracy and updated classification accuracy than the other two. Hoeffding Tree is



(a) Hoeffding Tree over 100 runs (b) Naive Bayes over 100 runs (c) SGD over 100 runs

Figure 4.1 – Maximum classification accuracy according to the amount of user feedback (positive & negative)

more likely to improve the classification accuracy, although SGD may increase the accuracy with less feedback. For M_{ground}, SGD does not show improvement on accuracy, while in most cases Hoeffding Tree provides higher accuracy than Naive Bayes after gathering feedback.



(a) Hoeffding Tree over 100 runs (b) Naive Bayes over 100 runs (c) SGD over 100 runs

Figure 4.2 – Maximum classification accuracy according to the amount of user feedback (only negative)

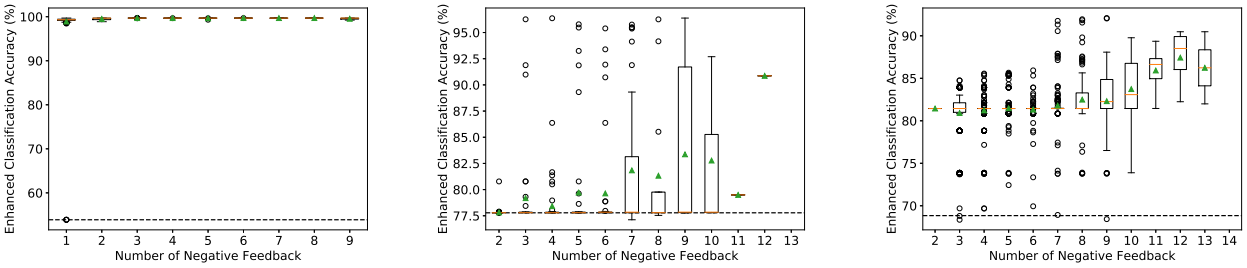
The experiments suggest that more positive feedback will be collected than negative feedback if the initial classification accuracy is above 50% and even more if the accuracy keeps increasing. Therefore, the negative feedback is always less requested, and thus the user is less prompted. Figure 4.2 analyses the maximum classification accuracy achieved when collecting only negative feedback. For all the algorithms, we observe that we can reach the highest classification accuracy with less than 15 negative feedback (half of positive and negative feedback). In more detail, for M_{pocket}, Hoeffding Tree reaches the highest accuracy instantly after one negative feedback, Naive Bayes does not change the accuracy at all and SGD requires more feedback to achieve the highest accuracy. For M_{door}, a similar phenomenon is observed: in most cases, Naive Bayes keeps the accuracy unchanged while both Hoeffding Tree and SGD increase the accuracy following negative feedback. For M_{ground}, SGD still does not provide any improvement on the classification accuracy, and in most cases, Hoeffding Tree provides a higher accuracy than Naive Bayes after receiving negative feedback. The result has shown that using only negative feedback can reduce the amount of feedback requests. Meanwhile, the increased classification accuracy has the same range as using both positive and negative feedback.

Overall, the Hoeffding Tree[81] is considered as the best candidate and used for the evaluation of the hierarchical inference and update approach detailed next. Also, we consider only negative feedback in our hierarchical approach. Note that the total number of feedback required needs be accumulated here, e.g. 15 for each and 45 in total, because the three classifiers are

updated individually using a single instance and three labels are requested every time.

Hierarchical feedback

Instead of requesting feedback for every classifiers on each instance, the hierarchical and negative feedback is provided to one of the three classifier. Our hierarchical approach is evaluated on $DATASET_2$, assessing the classification accuracy according to the amount of negative feedback. We performed 500 experimental runs.



(a) Accuracy is better 99% of the time for M_{pocket}

(b) Accuracy is better 86% of the time for M_{door}

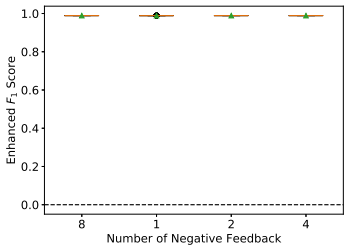
(c) Accuracy is better 71% of the time for M_{ground}

Figure 4.3 – Improvement of the binary H.Tree classification accuracy according to the number of (hierarchical and negative) feedback with $\lambda = 10$

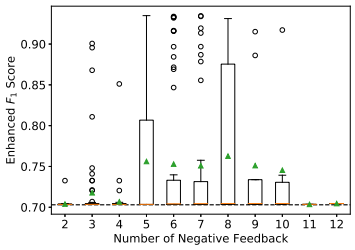
Figure 4.3 provides the classification accuracy according to the amount of (negative and hierarchical) feedback. Among the 13 (negative and hierarchical) feedback, at most 9 feedback are related to M_{pocket}, 12 to M_{door} and 13 to M_{ground}. The classification accuracy gets better 99%, 86% and 71% of the time for M_{pocket}, M_{door} and M_{ground}, respectively. In such a case, the mean classification accuracy for M_{pocket} and M_{ground} is quite high and much better comparing to the initial accuracy, while the mean enhanced accuracy for M_{door} is quite high, but is closer to the initial accuracy. While the accuracy in M_{pocket} remains stable, the classification accuracy for M_{door} and M_{ground} increases and then decreases a little before reaching maximum 91% and 90%, respectively. We undertook additional experiments that are not plotted here. These experiments showed that when the parameter λ is larger, the accuracy is more often better, but the runtime cost is also higher. For instance, with $\lambda = 15$, the accuracy is better 100%, 88% and 71% of the time for M_{pocket}, M_{door} and M_{ground}, respectively. Overall, requesting only negative feedback reduces the amount of feedback requested from the end user while the hierarchical approach requires even less feedback. Indeed, 13 (negative and hierarchical) feedback are requested rather than 90 (negative and positive, non hierarchical) feedback, or 45 (negative only, non hierarchical) feedback.

We perform 500 experiments so as to evaluate our hierarchical approach using the F_1 score, which is a measure of a test's accuracy considering both the precision and the recall of the test. The best (resp. worst) value attained by F_1 score is 1 (resp. 0). Precisely, the initial classifier trained with $DATASET_1$ is evaluated by simulating negative feedback, leveraging $DATASET_2$.

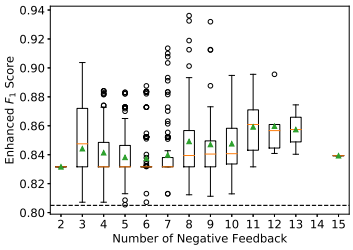
Figure 4.5 provides the F_1 score according to the number of (negative and hierarchical) feedback. Among the 15 (negative and hierarchical) feedback, at most 4 feedback are related to M_{pocket}, 12 to M_{door} and 15 to M_{ground}. The F_1 score gets an enhancement 100%, 90% and 71% of the time for M_{pocket}, M_{door} and M_{ground}, respectively. The enhancement of F_1 score is most significant for M_{pocket} and least significant for M_{door}. Overall, 8 hierarchical feedback can provide high F_1 scores for all the three classifiers.



(a) F_1 score is better 100% of the time for M_{pocket}



(b) F_1 score is better 90% of the time for M_{door}

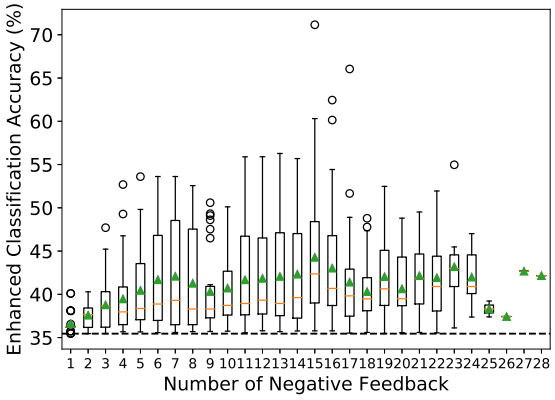


(c) F_1 score is better 71% of the time for M_{ground}

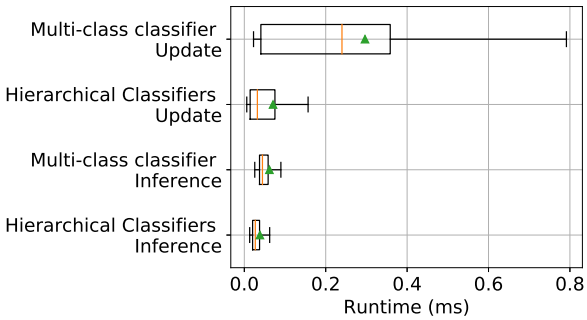
Figure 4.4 – Improvement of the binary H.Tree F_1 score according to the number of (hierarchical and negative) feedback with $\lambda = 10$

Hierarchical classifiers versus multi-class classifier

We can infer the context by either performing a single multi-class classification or by using three binary hierarchical classifications. The multi-class classifier distinguishes 8 classes as combination of in/out-pocket in/out-door and under/on-ground. To compare the multi-class classifier with our hierarchical classifiers solution, we perform 100 experiments (because many more updates are needed compared to the hierarchical way), using the same settings as in Section 4.5.1.



(a) Multi-class H.Tree classification accuracy according to the amount of (negative) feedback



(b) Single multi-class H.Tree versus hierarchical binary class H.Tree

Figure 4.5 – Improvement Multi-class H.Tree classification accuracy and Runtime for single multi-class H.Tree versus hierarchical binary class, with $\lambda = 10$

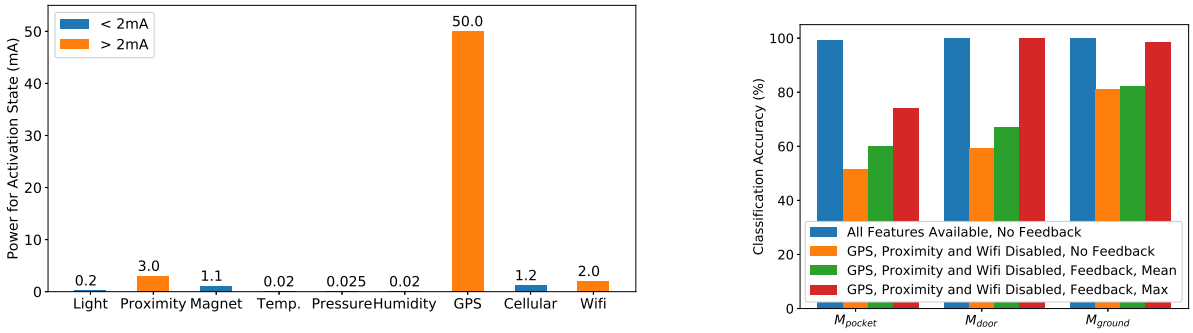
As illustrated in Figure 4.5a, the initial and enhanced classification accuracy of a multi-class classifier is much lower (around 2 times lower) compared to the ones of hierarchical classifiers (Figure 4.3). The enhanced accuracy is on average always lower than 45%. Besides, the multi-class classifier requires more feedback over all experiments (28 feedback are required over 30 feedback). Another drawback of the multi-class classifier is that the user has to select among 7 options as a feedback rather than answering "yes or no" to a simple and specific question.

Meanwhile, we performed 500 experiments to compare in Figure 4.5b the inference and update runtime of multi-class classifier and hierarchical classifiers. Our hierarchical solution is characterised by a much lower update runtime (median, mean and majority) compared to multi-class classifier. For inference, hierarchical classifiers induce a slightly lower runtime comparing

to the multi-class classifier because the hierarchical classifiers do not run all classifications every time.

Overall, the advantages of using a hierarchical classifiers (rather than a multi-class classifier) are many fold: (1) Using a classifier for each context element ensures that the classification accuracy remains high for each of them. (2) Each classifier relies on the vector of the most relevant features, resulting in a reduced inference and update runtime. (3) A classifier can easily be added/removed/replaced when a new context element needs to be managed for the benefit of the upper layer crowdsensing-based application. (4) Our hierarchical classification limits the number of classifications that is triggered, e.g., the in-/out-door context is assessed only if the device is outside the pocket. (5) The user feedback required for the personalisation of the hierarchical classifiers is simple and reduced.

Power consumption and accuracy



(a) Power consumption of the embedded sensors and communication modules

(b) Classification accuracy

Figure 4.6 – Classification accuracy and power consumption

Figure 4.6a presents the power consumption of the environmental sensors embedded in the Crosscall Trekker-X3 phone and of the networking modules[2], which together serve to gather the features. We observe that the light, temperature, pressure and humidity sensors are not power-consuming, contrary to GPS module, proximity sensor and WiFi component. In the following, we concentrate on evaluating the impact of disabling these three power-consuming features. We already considered the impact caused by the unavailability of environmental sensors in $DATASET_2$.

> $DATASET_3$**:** This dataset contains a training dataset ($20k$ instances) and a testing dataset ($20k$ instances): a Samsung Galaxy S4 GT-i9505 smart phone is used to collect $DATASET_3$ for the initial classifier and a modified version $DATASET'_3$ is used for evaluation. In $DATASET'_3$, the proximity sensor, the GPS module and the WiFi component are disabled on the device, thus the three features from them are unavailable.

Again, we simulated the hierarchical inference and update by 500 experimental runs. Figure 4.6b presents the classification accuracy for M_{pocket}, M_{door} and M_{ground}, when all the features are available, and when the GPS module, proximity sensor and WiFi component are disabled.

2. https://source.android.com/devices/tech/power/values.html

When the GPS module, proximity sensor and WiFi component are disabled, the classification accuracy diminishes to respectively 50%, 40% and 20%. After getting feedback, the mean classification accuracy increases by 10%, 8% and 1% respectively, while the maximum classification accuracy after feedback reaches 74%, 99% and 98% respectively. Overall, our approach personalises the classifiers and deals with the disabled power-consuming features according to user preference.

Our implementation requires $100MB$ of memory on a Nokia X6 Android 9 smartphone (Qualcomm Snapdragon 636) and leads to an increase of 5% of the CPU. The inference and update of contexts necessitates around $3MB$ of memory and an increase of 5% of the CPU. We evaluate the computation runtime of the entire inference and update phases. The inference runtime is on average $0.2ms$, $0.1ms$ and $0.1ms$ for M_{pocket}, M_{door} and M_{ground}, respectively, while the runtime necessary to update the model is $7.3ms$, $7.5ms$ and $10.0ms$ on average (with $\lambda = 10$).

4.5.2 Group-based Collaborative Crowdsensing

We evaluate the performance of our approach, assessing: (i) the impact on the power consumption of the device running the *BeTogether* middleware in the background, and (ii) the potential benefit of the collaborative crowdsensing at the edge from the standpoint of data quality and communication cost based on a one-year dataset obtained from the Ambiciti application for urban pollution monitoring. We note that due to privacy and commercial concerns, the Ambiciti company shared the data with us within the framework of a collaboration agreement while data cannot be released openly.

Power consumption

Table 4.6 – Active power of components for Nexus 5X

WiFi TX	WiFi Scan	Cellular TX	Light Sensor	GPS RX
173 mA	2 mA	186 mA	0.2 mA	60 mA

We estimate theoretically the power consumption on a single crowdsensor; Table 4.6 shows the power of the main components used in our crowdsensing middleware. The reference values are for the Google Nexus 5X smartphone, as provided by the Android OS (`https://source.android.com/devices/tech/power/values`) according to the power profile provided by the manufacturer. Herein, we select the light sensor, which involves a power consumption comparable to that of most sensors [52, 96]. Note that the power consumption associated with the transmission depends on the transmission duration, which increases linearly with the packet size. We assume that cellular and WiFi Direct communications have the same transmission speed.

Figure 4.7 shows the power consumption of a crowdsensor working individually at various sensing and upload frequencies. The upload is the most energy-demanding and the energy consumption can be reduced by lowering the upload frequency. As a comparison, Figure 4.8 provides the power consumption of the various nodes contributing to collaborative crowdsensing with regards to a high sensing frequency (every 1 minute) and upload frequency (every 10 minutes) for the individual case. Results show that the proxy always consumes the most energy due to the cellular transmission that takes place with the cloud, followed by the coordinator that communicates with the nearby devices to distribute the tasks. Other group participants

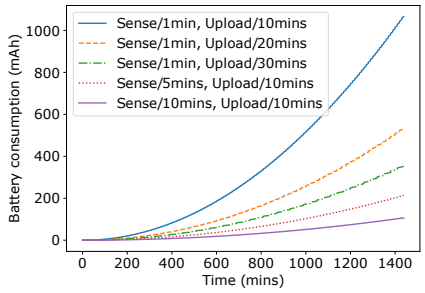


Figure 4.7 – Varying individual crowdsensing frequencies & Power consumption

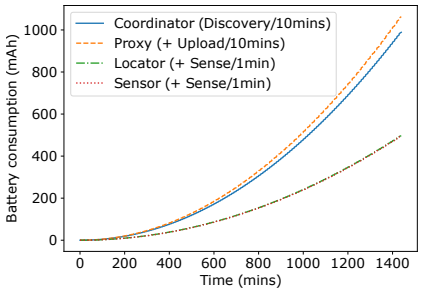


Figure 4.8 – Collaborative crowdsensing & Power consumption

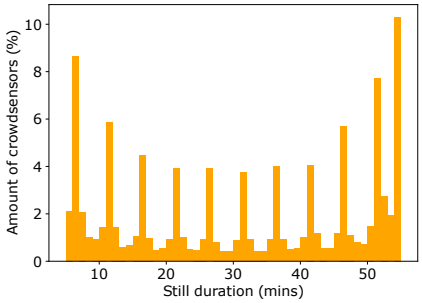


Figure 4.9 – Distribution of the still duration

consume much less energy compared to the individual case, even-though this consumption includes the cost related to discovery and D2D transmission to the coordinator.

Dataset-driven evaluation

Dataset We leverage a dataset produced by the Ambiciti (formerly called SoundCity) cloud-based crowdsensing application available on Google Play since 2015 and on Apple AppStore since 2016 (see `http://ambiciti.io`). Ambiciti monitors the noise pollution using the smartphone's microphone [167]. Our dataset contains 946,573 observations gathered both indoors and outdoors in Paris over the year 2017 from 550 crowdsensors, where the average uploading duty cycle is around 5 minutes. Each observation provides: the uploading time-stamp, the location and (anonymized) ID of the contributing device, the noise level and its measurement bias, a description of the user activity (still, on foot, on a bicycle, in a vehicle, unknown), and whether the device is in/out-pocket (based on proximity). With 550 crowdsensors for the all of Paris, the dataset is sparse. Hence, it does not provide the most suitable case for opportunistic collaboration at the edge. Still, this allows us to assess the effectiveness of *BeTogether*, even with a sparse deployment.

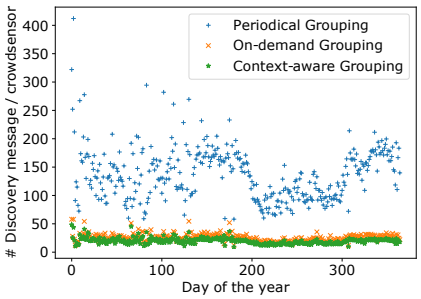


Figure 4.10 – Comparison of grouping strategies

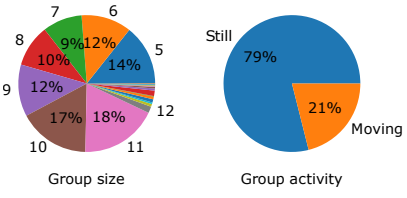


Figure 4.11 – Distribution of group properties

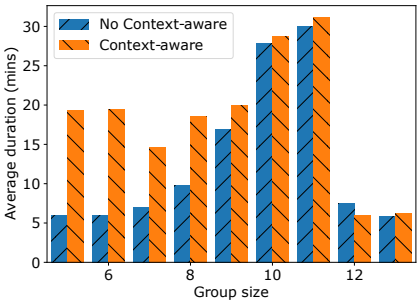


Figure 4.12 – Lifetimes of collaborative groups

Analyzing the crowdsensor behavior We first analyse the stability of the crowdsensors' context, as used for the configuration of groups, where we only consider the User Activity (UA). Indeed, the Physical Environment (PE) value is limited to the in-pocket case in the dataset, which only influences the sensor utility $u_s(i)$. The context is assessed daily. Starting with the initial location l of any crowdsensor i within the dataset, we consider that i changes group when

it reaches another location l' that is more than the D2D range away from l, and repeatedly so with l' as the new reference location.

Figure 4.9 then shows the distribution of the duration of the crowdsensor staying within the above estimated group for all the crowdsensors of our dataset according to the device's location: it varies from 10 minutes to 60 minutes where we recall that we set $D_{min} = 5$ minutes as the minimum duration of the group. Hence, many crowdsensors remain at the same location long enough to group.

Figure 4.10 further compares the three following grouping strategies in terms of the average number of messages sent per crowdsensor daily so as to discover nearby crowdsensors: *periodic* (after every upload) that is the approach found in related work, *on-demand* (detected by WiFi Direct), and *context-aware* that accounts for the crowdsensors' activities. In average, the amount of traffic generated by the on-demand strategy is 80.810% lower than the periodic approach, and the one of our context-aware grouping is 21.844% lower than the on-demand approach.

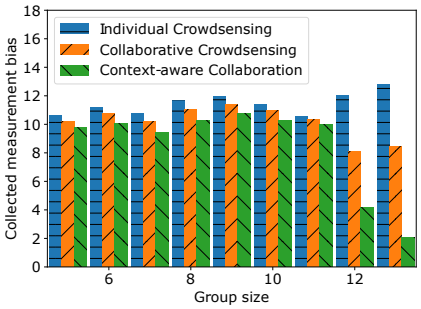


Figure 4.13 – Impact on collected data quality

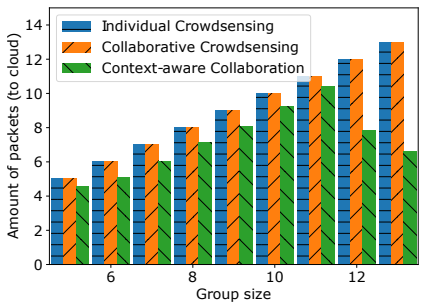


Figure 4.14 – Impact on uploaded data traffics

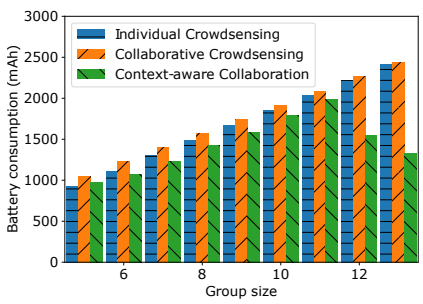


Figure 4.15 – Impact on overall battery consumption

Analyzing the efficiency gain of grouping In order to find the clusters of crowdsensors that are within D2D range (at 10m (re-scaled) away) from each other, we rely on the DBSCAN algorithm [58], which handles clusters that are arbitrarily shaped and that are of varying density. According to our parameter $\delta = 4$, that configures groups of size 5 and more, Figure 4.11 shows the distribution of the resulting group sizes in our dataset, with 79% of the groups being immobile.

Figure 4.12 further compares the average duration of the identified groups depending on whether the context is accounted for or not. Interestingly, results show that even in real life scenarios (including both immobile and mobile groups, not considering only still grouping as in Figure 4.9), our context-aware grouping finds groups of longer duration, which is up to 3.256 times of non context-aware grouping. As groups grow, the difference between a context-aware and non-context-aware approaches lowers because the likelihood of grouping co-located nodes having the same context gets higher. A decrease of lifetime is observed for groups of 12 members and more, which is partly due to the sparsity of our dataset and also the higher probability of members moving away from the group.

Although the crowdsensing data of our dataset is very sparse in time and space, the context-aware collaborative crowdsensing at the edge still brings benefit in terms of data quality and global resource consumption. It is especially efficient as the size of the group grows. Figure 4.13 shows that the *context-aware* collaborative approach delivers the best data quality: the collected measurement bias is reduced by up to 615% (resp. 407%) compared to an individual (resp. non-context-aware collaborative) crowdsensing approach. This is because of the selective sensing of *BeTogether* within each group, leading to the collection of the most accurate observations rather

than a simple average of the observations (non-context-aware collaborative crowdsensing) or than all the raw data (individual crowdsensing).

As shown in Figure 4.14, the amount of data that a collaborative and context-aware approach uploads to the cloud is reduced by up to 197% compared to both the individual and (non-context-aware) collaborative crowdsensing approach. There are two reasons for this: first, collaborative crowdsensing uploads only the aggregated data (i.e., concatenated hash tables) via the group proxy rather than uploading the raw data supplied by each crowdsensor (individual crowdsensing). Second, our context-aware collaboration also filters out the data that are of low-quality and that are collected in-pocket, which reduces the amount of data aggregated and uploaded to the cloud.

Finally, Figure 4.15 focuses on the power consumed per hour, which is associated with both WiFi Direct transmission (assuming that all the tasks are distributed) and cellular transmission: the power is reduced by up to 181% (resp. 183%) compared to individual (resp. non-context-aware collaborative) crowdsensing approach. Overall, these results show that the collaboration achieves a better data quality at a lower sensing and transmission cost when the group is larger because the tasks can be assigned to more devices, resulting in less task duplication and better context-aware filtering and aggregation.

4.5.3 Interpolation and Aggregation

In the following evaluation, the experiments are run either on a DELL Precision 7520 workstation - § 4.5.3 as a centralised server and § 4.4 for simulation, or on Android smartphones - § 4.5.3 as end-device testbed.

Accuracy metrics

We use the *Mean Absolute Percentage Error* (MAPE) and *Root Mean Square Error* (RMSE) to evaluate the accuracy of the interpolated and aggregated tensors. Given the observation data tensor $\mathcal{Y}$, the approximation data tensor $\hat{\mathcal{Y}}$ and the ground truth mask tensor $\mathcal{M}$ (indexing useful values), MAPE is defined as:

$$MAPE(\mathcal{Y}, \hat{\mathcal{Y}}, \mathcal{M}) := \frac{100\%}{\|\mathcal{M}\|^2} \sum \frac{\mathcal{M} * |\mathcal{Y} - \hat{\mathcal{Y}}|}{\mathcal{M} * \mathcal{Y}}$$

While RMSE is defined as:

$$RMSE(\mathcal{Y}, \hat{\mathcal{Y}}, \mathcal{M}) := \sqrt{\frac{\|\mathcal{M} * (\mathcal{Y} - \hat{\mathcal{Y}})\|^2}{\|\mathcal{M}\|^2}}$$

To compute MAPE and RMSE, we run 100 training rounds followed by tests. At each round, both training and evaluation sets are randomly shuffled, where: For the interpolation, 70% of the dataset is used for training (i.e., as actual observations to complete the tensor) and 30% to test (i.e., to assess the estimated values against the ground truth). For aggregation, the entire approximation tensor is used for the evaluation.

Interpolation evaluation

Figures 4.16 and 4.17 compare the robustness of the three inference strategies that are commonly used to interpolate physical phenomena: Ordinary kriging with Gaussian variogram model (*OK-Gaussian*), CP decomposition with Alternating Least Square (*CP-ALS*), and Gaussian Process

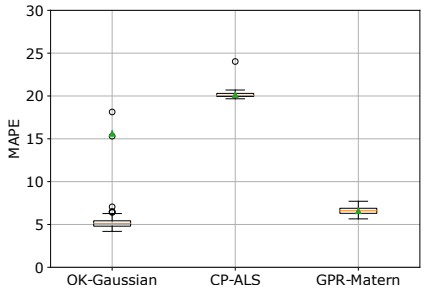


Figure 4.16 – Interpolation accuracy - MAPE

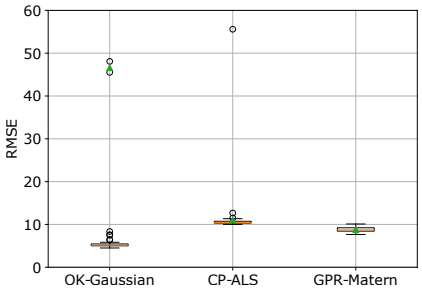


Figure 4.17 – Interpolation accuracy - RMSE

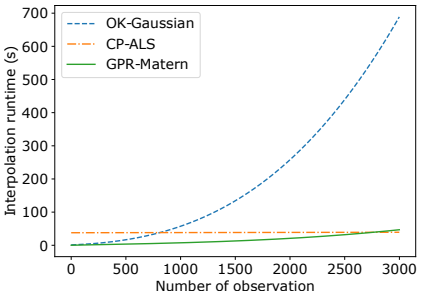


Figure 4.18 – Interpolation response time

Regression with Matern kernel (*GPR-Matern*). The interpolation is performed at the same PC (without involving any aggregation), using the dataset from which we selected on the day during which the largest amount of crowdsensing data were collected. The same experiments were run using the whole dataset and the same trends were observed. In the figures, the box corresponds to the interquartile range, the orange line is the median and the green triangle is the mean. At first sight, OK-Gaussian seems to be accurate and hence promising, given the low MAPE and low RMSE interquartile range and median. However, some wrong inferences lead to abnormal values as illustrated by high MAPE and RMSE mean values of 16% and 47, respectively. Similarly but to a lower extent, CP-ALS shows some abnormal RMSE. Instead, GPR-Matern provides both an accurate and robust inference: stable MAPE and RMSE without outliers –hence characterised by the lowest variance– is observed.

Figure 4.18 shows the response time of the three interpolation approaches. We run the experiments over the 365 days of our dataset, where the number of observations varies everyday. The response time of CP-ALS ($R = 1$) is constant in $\mathcal{O}(RIJK)$ regardless of the number of available observations since the computation applies on the entire fixed-size tensor. Both OK-Gaussian and GPR-Matern have a time complexity in $\mathcal{O}(n^3)$ with n being the number of observations used to fit the model. The response time of GPR-Matern is lower than OK-Gaussian, and below CP-ALS when the number of observations is less than 2800. Note that in our dataset, the number of observations collected by crowdsensors daily remains lower than 1500. Overall, GPR-Matern is the most efficient in terms of accuracy and robustness, while its response time is also relatively lower.

We further evaluated the efficiency of the three interpolation methods in terms of memory consumption. GPR-Matern consumes the least memory: around $3.114MB$, with a variance of 1.718. While the memory consumption associated with CP-ALS (resp. OK-Gaussian) is of $3.258MB$ with a variance of 0.422 (resp. $4.644MB$ with a variance of 1.437). Note that the memory consumption is stable and does not depend on the number of observations since our approach always uses a fixed-size tensor that is filled with zeros in the absence of observations.

Focusing on GPR, we assessed the accuracy (Figures 4.19 and 4.20) and associated response time (Figure 4.21) of the following kernels: constant, RBF, rational quadratic, and Matérn. The rational quadratic and Matérn kernels are the most accurate, while the former slightly outperforms the latter. However, the response time of the quadratic kernel is twice as much as that of the Matérn kernel. We therefore consider only GPR with Matérn kernel in our solution.

Aggregation simulation-based evaluation

The distributed aggregation encompasses several tasks: each crowdsensor converts the sensing data into a tensor, interpolates and potentially merges the tensors when meeting another.

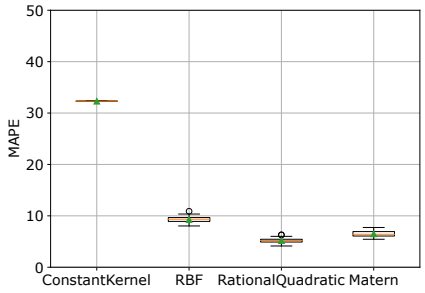


Figure 4.19 – Kernel accuracy - MAPE

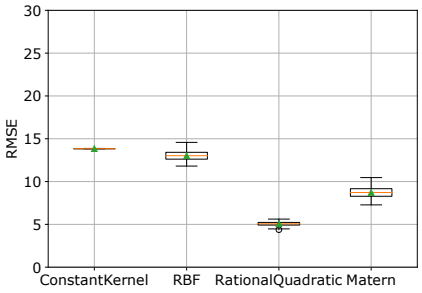


Figure 4.20 – Kernel accuracy - RMSE

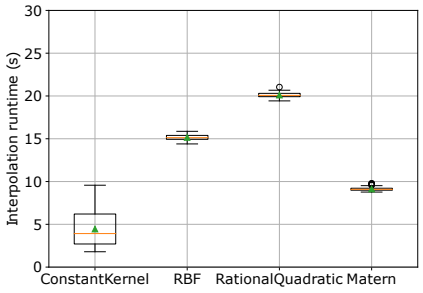


Figure 4.21 – Kernel response time

We assess the performance of the following distributed aggregation methods using our 1-year dataset:

- **Iterative aggregation** is a theoretical and sequential case in which the aggregation starts at the first crowdsensor that aggregates its tensor with the next crowdsensor and the aggregation process repeats with the following crowdsensors until the last crowdsensor is reached. This is the ideal case for which we ignore the actual locations of the crowdsensor.
- **Stochastic aggregation** represents the real-life scenario: an aggregation occurs when at least two crowdsensors meet as detected using the location and time proximity in the dataset. The aggregation process thus depends on the mobility of the contributing users. Upon a meeting, the merge base is selected randomly and $\beta = \beta' = 1$ for the generalised product-of-expert. Ultimately, all data are uploaded to and merged at the cloud, either directly or via a relay depending on the peer meetings.
- **Opportunistic aggregation** is similar to the above stochastic aggregation with the exception of the selection of the merge base and the chosen β values. Here, we set $\beta_s = 1.3$ and $\beta_{s'} = 0.7$ for crowdsensor s with lower D and crowdsensor s' with higher D respectively.

The above methods are further compared to the cloud-centric solution where the cloud server (DELL workstation in our experiment) collects the sensing data and performs the interpolation based on the whole dataset.

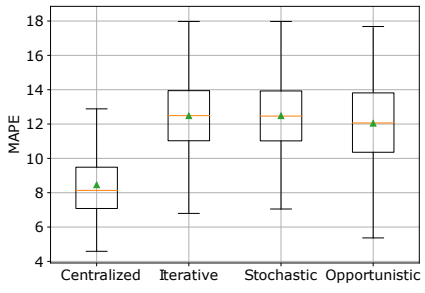


Figure 4.22 – Aggregation accuracy - MAPE

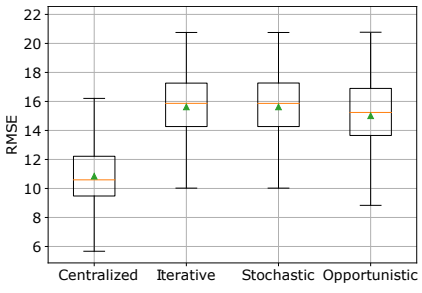


Figure 4.23 – Aggregation accuracy - RMSE

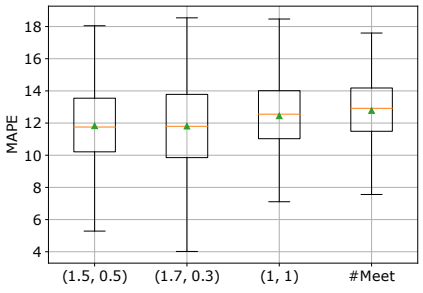


Figure 4.24 – Impact of β setting - MAPE

Figures 4.22 and 4.23 provide the MAPE and RMSE of the three above distributed aggregation approaches and of the centralized one. As expected, the centralized approach provides the most accurate inference: The MAPE mean equals 8.5% and the MAPE median is 8.1%;

the RMSE mean equals 10.8 and the RMSE median is 10.6. The accuracy of the distributed aggregations are quite similar, with a MAPE of around 12.5% with a median of 12.5% and a RMSE mean of 15.6 with a median of 15.8. In particular, our opportunistic aggregation is comparable to the centralised approach (e.g., the MAPE accuracy difference is 4% lower than the centralised approach). It has mean and median MAPE of 12.0%; a RMSE mean of 15.0 and a RMSE median of 15.2. Overall, our approach is characterised by very slight decrease of the accuracy compared to centralised aggregation.

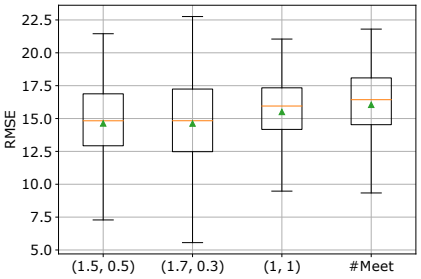


Figure 4.25 – Impact of β setting - RMSE

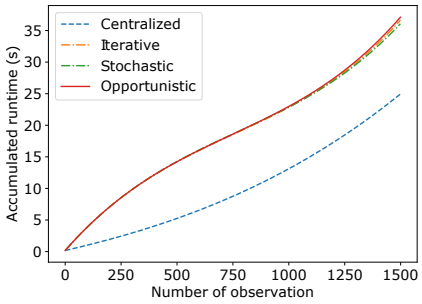


Figure 4.26 – Simulation response time

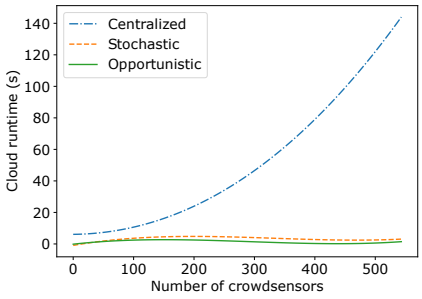


Figure 4.27 – Cloud server response time

Figures 4.24 and 4.25 compare different weighting configurations $(\beta_s, \beta_{s'})$ for crowdsensor s and s'. For three of the pairs, we manually set their values depending on the loss function D, i.e. the lower loss has a higher weight. For the fourth one, we set $\beta_s = 2\frac{n_s}{n_s+n_{s'}}$ using the ratio depending on the history number of aggregations n. The results show that assigning a higher weight to the crowdsensor with the lower loss function slightly increases the overall accuracy aggregated at the cloud.

Figure 4.26 shows the response time associated with the entire procedure including data pre-processing, interpolation and aggregation, with the simulation being run on the server. We observe that the centralised aggregation response time is lower than the accumulated response time of the distributed aggregation schemes. However, in the distributed approach, the cloud does not perform any interpolation, thus the centralised interpolation requiring a high response time is eliminated. As illustrated in Figure 4.27, the response time associated with the aggregation at the cloud server significantly increases when the number of crowdsensors gets high. Instead, when the interpolation and aggregation are mainly performed by crowdsensors, the cloud response time is almost negligible regardless of the number of crowdsensors. In addition, the storage requirement is minimised on cloud because the data tensor size is always unchanged when aggregating new incoming data.

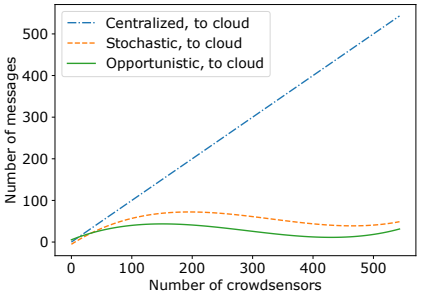


Figure 4.28 – Directly uploaded messages

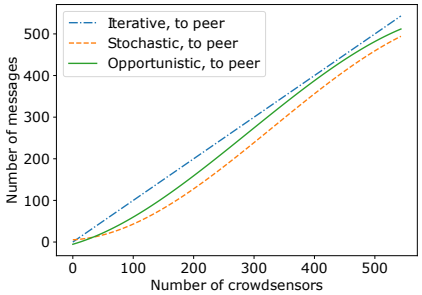


Figure 4.29 – P2P forwarded messages

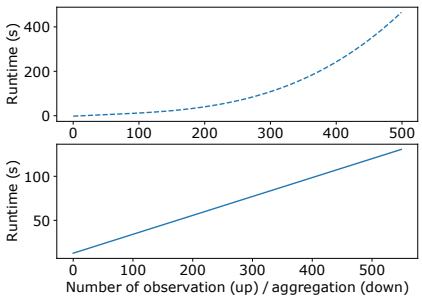


Figure 4.30 – On-device response time

Figures 4.28 and 4.29 compare the amount of traffic uploaded to the cloud and relayed among the crowdsensors. The traffic is evaluated in terms of the number of actual aggregations. As expected, distributed aggregation reduces the amount of traffic uploaded to the cloud and hence the cellular network occupancy is kept to a minimum. In particular, our opportunistic aggregation drastically reduces the cloud uploading by 54.2% compared to the stochastic aggregation. A portion of the traffic sent to cloud is replaced by the traffic forwarding among crowdsensors; there are more aggregations and thus more P2P messages generated when the number of crowdsensors increases. This result also validates our estimation that crowdsensors with better inference quality tend to have more meetings and thus relay opportunities.

Aggregation testbed-based evaluation

Using Android smartphones, we empirically assess the performance associated with the prototype in terms of response time and energy consumption. We conduct the experiment using our 1-year dataset as data input.

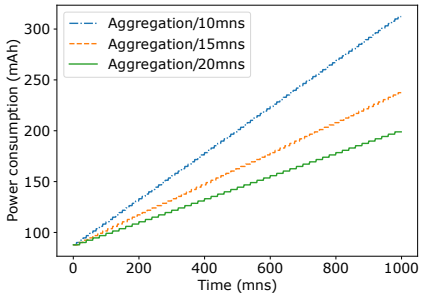


Figure 4.31 – Energy consumed

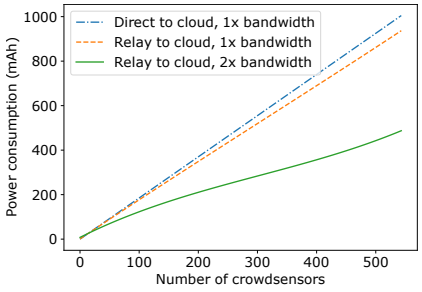


Figure 4.32 – Energy consumed by network

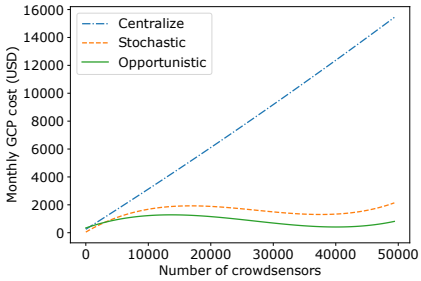


Figure 4.33 – Cloud monthly financial cost

Figure 4.30 shows the interpolation response time depending on the number of observations, and the aggregation response time depending on the number of aggregations, where the experiment is run on a SAMSUNG GALAXY S7 smartphone. Note that the figure shows no more than 500 entries, which is in practice a very high number of observations collected by a crowdsensor daily. As expected, interpolation is computationally intensive compared to aggregation, whose response time is comparatively negligible: the interpolation takes a couple of minutes when the number of observations is greater than 500, while the aggregation takes less than 100 seconds for a number of aggregations below 500 and for a number of observations per crowdsensor varying from 1 to 500. When we consider 500 observations, the interpolation consumes the most energy with 88mAh, and the aggregation consumes only 2mAh. Figure 4.31 shows the energy consumed by a smartphone that implements our opportunistic aggregation when the P2P meeting frequency varies: the more frequent is the aggregation, the more energy is consumed. Recall that each crowdsensor executes the interpolation only once and it can be piggyback, e.g., executed when the smartphone is charged during night. Nevertheless, in practice, the related energy consumption gets limited because the crowdsensor usually has already relayed/aggregated its data before meeting ≈ 10 crowdsensors for a single day.

In order to evaluate the power consumption due to communication, we rely on the power profile of a LG NEXUS 5X smartphone provided by the manufacturer. Assuming a constant bandwidth, the WiFi transmission consumes 1.72mAh while the cellular transmission consumes 1.85mAh power. We estimate the energy consumption associated with all crowdsensors. Figure 4.32 shows the energy consumption associated with the local P2P traffic as well as the cellular Internet traffic. The energy is reduced by replacing uploads with P2P relays. When the P2P

bandwidth is higher than the cellular bandwidth (e.g., two times in figure), the advantage is more significant. Furthermore, the local P2P traffic is not only less costly in terms of energy aspect but also in terms of budget.

Impact on the financial cost

We estimate the financial cost associated with our crowdsensing system, using the Google Cloud Platform (GCP, `https://cloud.google.com/products/`) as an example of cloud platform. GCP provides the following key services: *Cloud IoT Core* is responsible for connecting the cloud to the IoT devices and establishing a two-ways communication. Upon the reception of a packet, the *Cloud Pub/Sub* service creates and delivers an event notification to *Cloud Functions* that implements basic operations (e.g., average, maximum, minimum) needed to pre-process the data. *BigQuery* is used to temporarily store the pre-processed data during the aggregation and interpolation. *Compute Engine* runs a virtual machine that performs the advanced computation (e.g., interpolation and aggregation). *Cloud Bigtable* is a NoSQL database that stores the resulting aggregated and interpolated sensing data.

The use of each of the above services is priced: for a detailed description, one may refer to `https://cloud.google.com/pricing/list`. In a nutshell, the price depends on the amount of network traffic received/sent, the amount of storage needed, and the load associated with the computation (e.g., number of pre-processing functions invoked and response time associated with the interpolation and aggregation). Figure 4.33 shows the monthly financial cost associated with running different crowdsensing in GCP, assuming that each crowdsensor sends a $2MB$ packet everyday. The cost associated with the centralised approach increases linearly because the number of uploads and the computation involved to interpolate the phenomena are both high. Instead, the costs of the stochastic and opportunistic approaches remain low because communication toward the cloud is reduced and only lightweight aggregation is performed at the cloud. Our opportunistic aggregation outperforms the stochastic approach because we further reduce the computing load on the cloud.

4.6 Conclusion

One of the major benefits of crowdsensing is the possibility to monitor environmental phenomena at an urban scale, simply leveraging the abundance and capacity of people's smartphones. However, as the number of contributors grows, the increasing number of observations that the crowdsensing systems must process gets challenging: the high network and financial cost associated to a cloud-centric system hinders the widespread deployment of crowdsensing, and the high computational cost due to the large amount of data makes the modelling of the environmental phenomenon intractable.

We tackle the above issues by exploiting the increasing computing capacity of today's smartphones, that is, we enforce a collaboration strategy at the edge so as to enhance the quality of the data transferred to the cloud while reducing the related communication cost and resource consumption. For this purpose, we introduce a set of utility functions that assess to which extent a device should carry out a given crowdsensing task, while achieving a trade-off between the benefit for all (for the whole group) and the related cost for the device.

Furthermore, we distribute the interpolation and aggregation associated with the sensing data at the powerful end devices. To do so, we introduce a middleware that runs on the smartphone to capture complex and potentially non-linear relationships among the collected observations across both space and time by relying on Gaussian Process Regression and 3D tensors.

Then, the resulting tensors are opportunistically combined together following a stochastic process based on the physical encounters of people. Rather than applying a naive combination (e.g., averaging) of the tensors that would actually degrade the quality of the sensing data, our solution performs a Product-of-Expert on the inference. The benefit our approach is threefold: (i) a selected crowdsensor (i.e., expert) may independently establish a very precise interpolation of the regions covered by a crowdsensor group; (ii) the aggregation resulting from the Product-of-Experts is sharper than any of the individual tensor and renders much more tractable the establishment of the overall tensor; and (iii) the computation achieved on the device is energy-efficient. Indeed, the evaluation using simulation and prototype implementation, together with a real-world dataset shows that our approach significantly reduces the transmission to, and the computing resource consumed on, the infrastructure server, without compromising the overall data accuracy.

Chapter 5

Information centric Networking

5.1 Context and Motivation

IoT is an ever growing ecosystem in which the *Things* (*e.g.*, smartphones, sensors, actuators, RFIDs, just to name a few) continuously exchange some data streams with other interested *Things*, applications and users spread across the Internet. IoT is an event driven ecosystem in the sense that reactions (*e.g.*, actuations, decisions) are triggered after an event or a series of events is captured by the event producer (*e.g.*, a sensor). Events correspond to some changes in the state (*e.g.*, a temperature increase, an excessive sound level) or simply an update. Relying on event notification permits to minimise the need of integration among *Things* and applications. As a result, *Things* are easily deployed and operate independently, as needed.

The IoT presents a number of challenges, especially in regard to routing and aggregation of a continuous flow of information that is exchanged among things. Thus, our attempt is to support an efficient communication between Things/users across heterogeneous networks by adopting an information-centric philosophy that heralds a new communication style, in which the data flow is governed by both the Things/users interests and the data content. More specifically, we build upon the publish/subscribe paradigm in which event producers send notifications to subscribers who have previously expressed their interests. The degree of expressiveness, which permits the users to describe their interest, is of critical importance and greatly differs from one publish/subscribe system to another. This leads to the categorisation of these systems into the two following classes: *Topic-* versus *content-based publish/subscribe systems*.

With *topic-based pub/sub systems* [180, 158, 159, 119, 67, 178, 35], a subscriber identifies a topic of interest (i.e., a keyword) and is thereby endowed with a low level of expressiveness. Each individual topic can be viewed, and thereby abstracted, as a separate channel /pipe, which is further mapped to a interested group of subscribers towards whom a notification is forwarded. Partitioning the topic space - and thereby the event space - into sub-topics leads to the creation of separate groups in which subscribing is equivalent to becoming a group member, with one group defined per (sub)topic. Due to the simplicity and ease of use of topic-strings, topic-based pub/sub systems became widely employed as part of enterprise and message-oriented middlewares [158, 159, 109, 40] and are now the cornerstone of modern stream analytic pipelines [71, 153]. Still, despite their effectiveness topic-base pub/sub systems suffer from a lack of expressiveness, meaning that that many irrelevant notifications are disseminated.

Content-based publish/subscribe systems overcome the limitations of topic-based publish/ subscribe system by improving the expressiveness of subscriptions. A subscription is composed of a set of filters (*e.g.*, "String function = temperature sensing", "String unit = Celsius", "Integer priority $>$ 1") that are further used to match event notifications. Thus a subscriber can easily combine several filters applying to notification attributes [34, 155, 13, 84, 124]. However this expressive subscription process comes with a more sophisticated and fine-grained filtering, which increases the broker workload. Indeed, the proposed filtering algorithms (§5.2) scale poorly (*i.e.*, sublinearly) with an increase of the event notification size (*i.e.*, number of attributes) and the

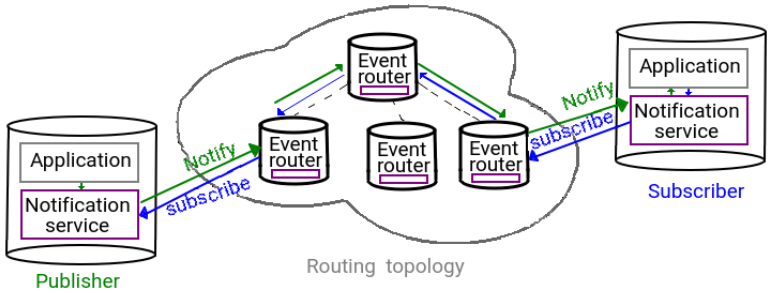


Figure 5.1 – Overview of the routing structure over which routers propagate the subscriptions and event notifications

subscription size (*i.e.*, number of filters), not to mention the volume of event notifications. Thus many of the approaches to scale the Publish/Subscribe system share the overheads of subscription management (matching, update) and event filtering among distributed routers. Whilst a variety of distributed notification systems have been proposed, a significant portion [34, 16] assume a pre-established organisation, which is not a realistic model for large-scale and geographically distributed networks.

Within this work we focus our attention on content-based publish/subscribe systems. Our aim is to render our content-based publish/subscribe system accessible by/from any *Things* over a wide-area network while overcoming some of the limitations of distributed content-based publish/subscribe systems, including: (i) the lack of robustness and flexibility [70], and (ii) the computing-gluttony associated with the fine-grained filtering. Our primary concern is hence to define – from design to the implementation – an effective strategy for dynamically organising the delivery of event notifications so that it scales and tolerates failures. Another key goal is to lighten the filtering process that takes place during the event notification delivery, without compromising the expressiveness of the subscriptions and the fine-grained nature of the filtering.

The publish subscribe system we propose is non-centralised so as to avoid using a central access point that may become a bottleneck. The system self-organises in a cluster-based hierarchical structure that self-adjusts upon link/device failures and device arrival and departure while ensuring strict control over the underlying structure. Conceptually, cluster heads serve as routers (Figure 5.1) and forward event notifications (as well as (un)subscriptions) to the next routers, towards their final destination(s). The advantage of a hierarchical structure is threefold. First, the cluster-based structure enables a short and situation-aware control loop, in which cluster members (a.k.a *Things*) can individually adapt and react in real-time to local events. Second, the hierarchical structure supports the aggregation [115], subsumption [85] and correlation of event notifications and hence alleviates the traffic load. Higher-level decisions can also be made on the basis of a macro-view/ state through an extended and hierarchical control loop.

The key activity of a router is to convey the event notifications and the related (un)subscriptions. In order to filter event notifications and (un)subscriptions (§ 5.3) the router maintains a repository containing the received subscriptions. It is noteworthy that most routers get unnecessarily overloaded because they perform the same filtering task *i.e.*, each router goes through the repository looking for the same subscription(s) (*a.k.a.* filter(s) to apply) as its neighbours. In order to alleviate the filtering process we introduce (§ 5.4) a new, compact way of representing filters that must be applied and that form a subscription. We rely on the Bloom filter [25], which corresponds to a compact approximation supporting probabilistic membership queries. Bloom filters intuitively introduce small space overheads when stored and conveyed due to their compact nature. In addition, Bloom filters save a substantial amount of time performing

the membership test that takes place during the notification/(un)subscription forwarding. The probabilistic nature of the Bloom filter means that false positives may eventually occur while there are no false negatives.

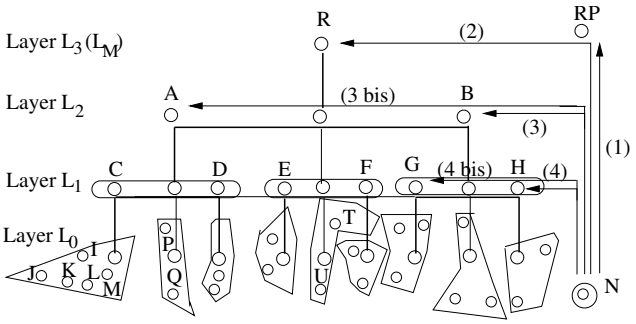
This constitutes a classical trade-off between space (*i.e.*, storage and communication), time (*i.e.*,, computation) and accuracy (false positives): increasing the performance of one of the three factors is done at the expense of the others. We formalise the problem of approximating and summarising the subscription filters into Bloom filters as an optimisation problem (§ 5.5). In particular we determine the degree of approximation desired and thereby a suitable size of Bloom filter so that the probabilities of false positive/negative always remains negligible. We further introduce a lossy compression algorithm that conveniently compresses the Bloom filter when needed. As a complement, we aim at improving the search algorithm that is used by a router looking for the applicable subscription(s)/filter(s) to filter the incoming notification. As shown during our evaluation (§ 5.6), the space cost associated with the indexing, is compensated by the gain in terms of responsiveness achieved when forwarding event notifications and (un)subscriptions. Scalability and expressiveness are two conflicting goals that a content-based publish/subscribe faces: the more expressive the subscription, the more overloaded the router is by the forwarding process.

5.2 Group-based Publish-Subscribe System

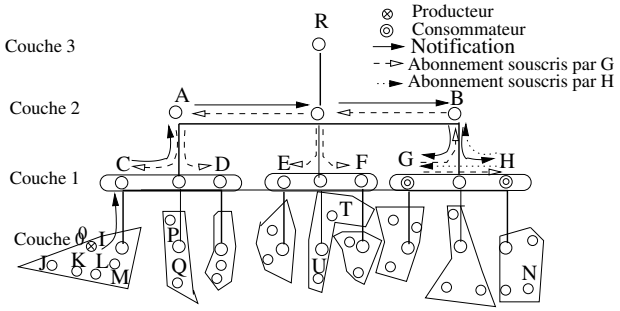
We adopt an event communication model that derives from the well-known asynchronous publish/subscribe paradigm in which consumers express their demands to producers during a subscription process and producers transfer to subscribers the description of any event that has been triggered locally. From a communication standpoint, our distributed event notification system exchanges notifications and control messages (i.e., (un)subscriptions) between producers and subscribers through a collection of intermediate routers that host our notification service. The key objective pursued by any router lies in forwarding a notification to a router only if there exists a consumer interested in receiving it. For the purpose of selectively forwarding notifications, each router holds a repository that includes the received subscription along with a list of the routers, which forwarded it. Indeed, a neighbouring router constitutes the potential candidate towards which the corresponding notifications are forwarded. Intermediate routers are organised into a cluster-based structure (also called an overlay) in which each cluster leader (a.k.a, router) keeps the information concerning: (i) the cluster members under its responsibility and (ii) its connections with other cluster leaders. Such a cluster-based structure facilitates the aggregation and the correlation of notifications and as a result of this significantly reduces bandwidth usage.

5.2.1 Underlying Group Communication

In order to dynamically manage the delivery structure to react upon any network failure, we rely on the Nice protocol [17], which is an application-level protocol originally developed to support video streaming over the Internet. The Nice protocol creates a self-organised, cluster-based hierarchy of n_l layers (n_l=$log(n_n)$, with n_n designating the number of nodes that are expected to join the group, in which each layer is composed of a set of bounded-size clusters controlled by a cluster head. The reason for setting bounds on the number of layers and on the cluster size is twofold. First, it ensures a control overhead of about $log(n_n)$ at each node. Second, it bounds the length of the path used for delivering notifications, and thereby the related delay ($o\ log(n_n)$). In practice, to warrant a loop-free structure, each node belongs to the lowest layer



(a) Arrival of a new node N: N contacts the RP and the closest cluster-heads of each layers so as to join the closest cluster at layer 0



(b) Message Routing: G and then H send an identical subscription to their cluster-head B. G's subscription is disseminated across the cluster-based structure. Later on, I generates a notification, which matches G's and H's subscription and is forwarded to G and H.

Figure 5.2 – Routers organised into a cluster-based structure through which messages are transmitted

(L_0) and only the leader of a cluster located in a layer L_i belongs to the upper layer L_{i+1}. A node keeps information restricted to its cluster(s) and each member periodically sends a *keep-alive* message. Thus, a cluster leader may passively detect a node failure and initiate a cluster merge/split if the cluster size differs significantly from the desired bound. The Nice protocol relies on a Rendezvous Point that provides bootstrapping information to any new node that joins the cluster-based structure. The addition of a new node N to the structure is handled as follows (Figure 5.2a): N contacts the Rendezvous Point (RP), which provides bootstrapping information relating to (i) the group configuration (e.g. the identity of the root R, number of layers, the cluster's bounded-size) and (ii) the event system configuration (e.g. the number of hash functions, hashing algorithm in use). Next, with the bootstrapping information given by RP, N selects the closest cluster head (B) and refines progressively its selection at each layer so as to ultimately join the closest cluster in layer L_0 (i.e., H's cluster in Figure 5.2a). Once created, the resulting delivery structure is used to propagate messages generated by the publish/subscribe system.

5.2.2 Cluster-based Routing

The main challenge related to routing over the cluster-based structure stems from the need to minimise the traffic and the computational load of routers. This calls for:

1. Duplicating messages as close as possible to their respective consumers, while filtering notifications as close as possible to event producers.
2. Minimising the number of control messages that are propagated.
3. Minimising the processing load induced by filtering notifications.

In the following, we describe to which extent our routing strategy meets the above commitments.

Subscription Routing. In order to propagate less subscriptions, a router determines if a subscription should be forwarded or not, and if so, towards which neighbouring router(s). To clarify this decision making we consider two consumers, G and H (Figure 5.2b), which declare their interests in receiving notifications for event e. Let us assume that G first sends a subscription to its neighbouring routers B and H and that shortly after H sends an identical

subscription to its neighbouring routers B and G. Upon receiving G's subscription, B forwards it to its upper-layer router R and its sibling A. Following, G's subscription is forwarded to all the remaining routers: R sends G's subscription to E and F while A forwards G's subscription to C and D.

Later, upon receiving H's subscription, B does not propagate it to R and G, because there is no need for B to propagate a subscription for a notification that it has already previously received. Herein H's subscription is not propagated because an identical subscription (G's subscription) was already propagated. The same propagation process would apply if H's subscription were covered by G's subscription since this would mean that the events that match G's subscription(s) also match H's subscription. Subscription is routed as follows: if the subscription is ascendant[1], then that subscription is forwarded[2] to the upper-level router. Conversely, if the incoming subscription corresponds to a descendant message or has been issued by a clustermate, then that subscription is forwarded to all lower-level routers. The unsubscription routing is similar to the subscription routing: a router forwards an unsubscription only if the related subscription was previously forwarded to that router.

Event Notification Routing

Figure 5.2b illustrates our presentation on notification routing: an event producer I generates a notification e in which two consumers are interested (G and H). I then sends a notification e to its cluster head C, which propagates it to A since a subscription that matches e was previously issued by A. Following, A propagates the event notification to B, which in turn propagates it to G and H. Overall, the event notification is not propagated to all the routers: any router that receives the notification selects the router(s) towards which the notification should be propagated. The selection process consists in first matching the notification against the subscription summary and then in extracting the list of neighbouring routers that should forward the matching subscriptions. The propagation of the notification over the delivery structure formed by the router is the same to the one carried to disseminate (un-)subscriptions.

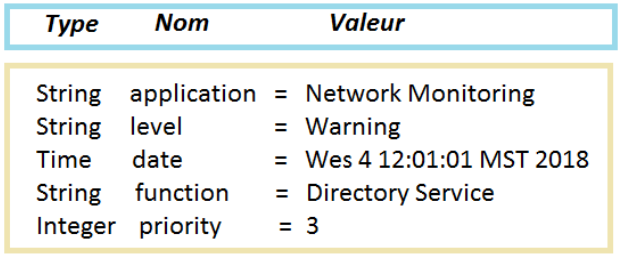
5.3 Notification and Subscription Representation

The effectiveness of the routing depends on the structure used to represent notifications, (un)subscriptions (§5.3.1 and §5.3.2) and on the structure used to organise all subscriptions (§5.4).

5.3.1 Notification and Subscription Formats

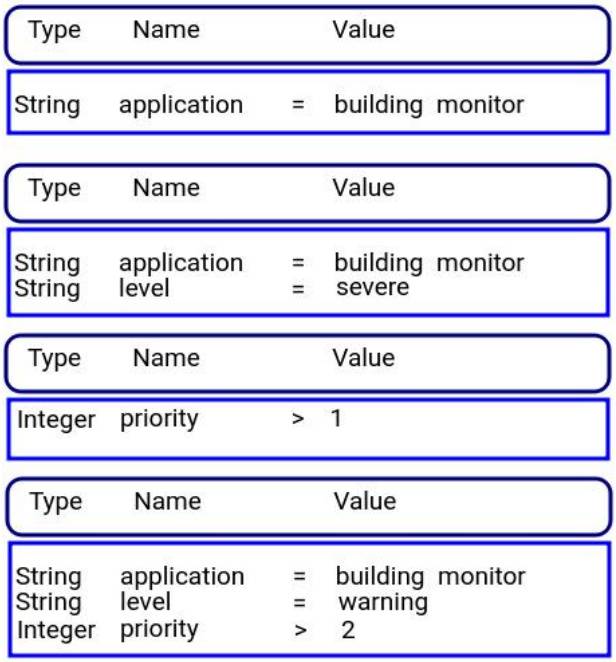
An event notification (or simply a notification) is composed of a set of n_e attributes that are typed and that describe the event (Figure 5.3a); each attribute α_i (with $1 \leq i \leq n_e$) consists of a type as well as a name-value pair: $\alpha_i = (type_{\alpha_i}, name_{\alpha_i}, value_{\alpha_i})$. An attribute type belongs to a predefined set of primitive types(e.g., string, integer or date). Figure 5.3a illustrates a notification issued by a network-monitoring application that informs about a directory service being stopped.

1. An *ascendant subscription* is conveyed to an upper-level router.
2. If we consider an edge router belonging to layer L_0 (resp. L_M), the subscription is not forwarded to a lower (resp. upper) layer.



Type	Nom		Valeur
String	application	=	Network Monitoring
String	level	=	Warning
Time	date	=	Wes 4 12:01:01 MST 2018
String	function	=	Directory Service
Integer	priority	=	3

(a) A notification is generated by a monitoring application. The notification is composed of a set of attributes.



Type	Name		Value
String	application	=	building monitor

Type	Name		Value
String	application	=	building monitor
String	level	=	severe

Type	Name		Value
Integer	priority	>	1

Type	Name		Value
String	application	=	building monitor
String	level	=	warning
Integer	priority	>	2

(b) Four different subscriptions. Each subscription is made of a set/conjunction of filters ; each filter is characterised by a type, name, operator and value.

Figure 5.3 – Notification and subscription

A subscription refers to a conjunction[3] of n_a filters $(f_1, \cdots, f_j, \cdots, f_{n_a})$ where each filter f_j designates a consumer's interest for a particular event. Each filter f_j is characterised by a type, a name, a predicate operator (e.g. $=$, $\leq$, `substring`) and a value. We say that an attribute $\alpha_i = (type_{\alpha_i}, name_{\alpha_i}, value_{\alpha_i})$ is covered by a filter $f_j = (type_{f_j}, name_{f_j}, operator_{f_j}, value_{f_j})$ iff $type_{\alpha_i} = type_{f_j}$ and $name_{\alpha_i} = name_{f_j}$ and $operator_{f_j}(value_{\alpha_i}, value_{f_j}) = true$. The notation $f_j \subset \alpha_i$ means that the filter f_j matches / is covered by the attribute α_i. More generally, an event notification $e = (\alpha_1, \cdots, \alpha_{n_e})$ matches a subscription $a = (f_1, , \cdots, f_{n_a})$ if the subscription matches all the filters constituting the subscription: $a \subset n$. Among the four subscriptions in Figure 5.3b, only the second subscription matches the notification of Figure 5.3a.

Similarly, a subscription a' is covered by a subscription a ($a' \subset a$) if all the filters constituting a' are covered by those of a. This also implies that any event notification matching a also matches a': $\forall e / a' \subset e \Rightarrow a \subset e$. Note that the covering relations is:

- reflexive: $a \subset a$ is always met.
- anti-symmetric: if $a \subset a'$ and $a' \subset a$ then $a = a'$.
- transitive: if $a \subset a'$ and $a' \subset a''$ then $a \subset a''$.

The establishment of a covering relationship constitutes a resource-consuming process that reiterates upon the receipt of a (un)subscription or of an event notification. In both cases a router attempts to discover if there exists (at least one) subscription s that covers the (un)subscription s' or that matches the event notification e. To this end, the router should go through all n'_s filters of the (un)subscription or all the n_e attributes of the notification e so

3. A disjunction is obtained using several subscriptions.

as to assess if the attributes are covered by or match the filters constituting the subscription s. This results in $O(n_s.n_{s'})$ or $O(n_e.n_a)$ comparisons if the attributes and filters are not ordered, and in $O(min(n_e, n_s))$ (resp. $O(min(n_s, n_{s'}))$ comparisons if they are ordered. Overall, the handling of a new (un)subscription or event notification is laborious with content-based pub/sub systems while topic-based publish/subscribe systems filter events and handle (un)subscriptions in $O(1)$ when the subscriptions are stored in hash tables.

5.3.2 Subscription Organisation - State of the Art

Few solutions [112, 125, 34, 24, 85, 32] thus far address reorganisation of subscriptions so as to improve the notification filtering and the (un)subscription process. All propose to organise the subscriptions into a single data structure, which corresponds to a list, a partially ordered set (or poset), a R-tree, a B^+-tree or a binary decision diagram. As detailed in the following, performance associated with notification filtering or with the establishment of a new (un)subscription covering varies depending on the data structure:

- **List** [112, 125] - In a list containing n subscriptions, the establishment of the covering relationship with an (un)subscription s' is costly: the covering is established in $O(n.n_s.n_{s'})$ if subscriptions are not ordered and in $O(log(n).min(n_s, n_{s'})$ if ordered. The filtering of an incoming event notification e takes $O(n.n_e.n_s)$ when subscriptions are not ordered and $O(log(n).min(n_e, n_s))$ if subscriptions are ordered. With a list, the subscriptions are stored one by one. The alternative structures described below, combine the filters that form the subscription into a single data structure, which is organised in different ways.

- **Poset** [34] (see Figure 5.5 on page 103) exploits the covering relations of a set of filters by applying a partial order on the set of subscriptions. Thus, any subscription at the upper level covers its descendant (Figure 5.6a). In practice, a poset corresponds to a direct acyclic graph whose size has an information-theoric lower bound of $n_p^2/4 + O(n_p)$ [113] and a (worst case) space complexity of $O(n_p^2)$ [34] with n_p defining the number of elements stored in the poset. Adding/removing a subscription or filtering an event notification necessitates the lookup of subscription(s), which is done [34] by traversing the poset following a breadth-first order and starting from the root. This lookup has a worst case complexity of n_p, though in practice it may be lower. This results in a subscription process done in $O({n_p}^2)$, an unsubscription process of $O(n_p^3)$, and a notification matching of $O(n_p)$. The two following data structure (R-tree and B^+-tree) also exploit the covering relationship.

- **R-tree** [24] has been traditionally used to support spatial queries. An R-tree is a tree in which each internal node has between m and M children. A node refers to the smallest poly-space rectangle (a.k.a set of filters) that encloses all the rectangles of its subtree (a.k.a covers all its descendants). A range query is answered thought a top-down traversal, starting at the root and recursively following the subtree of each matching child. The height of an R-tree containing n_r nodes is $\lceil log_M(n_r) \rceil - 1$ with n_r corresponding to the number of nodes. However, there is no acceptable guarantee, given that more than one sub tree under a node may need to be visited. In the worst case, the lookup with an R-tree is as expensive as a naive sequential scan.

- B^+-tree is an ordered tree wherein all the intervals (all the filters) are stored and sorted at leaf level for fast traversing. A B^+-tree is designed to have a fairly high number of children for each internal node so that the height of the tree remains relatively small.

After insertion and deletion, a B$^+$-tree may thus require balancing. In a tree of size n, performing a range query with k elements occurring within the range requires $O(log_b n+k)$ operations.

- **Binary Diagram** [32] structure represents a generic data structure used to represent the Boolean functions constituting a filter. A binary decision diagram may be represented as a DAG with a unique root, in which the internal nodes correspond to the Boolean functions. Thus, there are exactly two outgoing edges, labelled 0 or 1, for each inner node. With the binary decision diagram, an arbitrarily complex Boolean expression of n variables can be straightforwardly evaluated, by simply descending the DAG in n steps according to the values of the n Boolean functions. In counterpart, the size of the tree may exponentially grow with the number of functions.

Overall, the list, poset, R-tree, B$^+$-tree and the binary decision diagram all place a high load on routers and lead to a lower throughput and higher end-to-end latency.

Overload Reduction

The cost incurred by a lookup may be reduced by keeping the number of nodes in the data structure to a minimum , thereby ensuring a faster traversal. This can be achieved in three ways: (i) reordering the data structure, (ii) merging the overlapping filters [111, 41, 24] or (iii) exploiting the fact [162] that there is no need to store the (new) subscription(s) that are already covered by other subscription(s). A (re)ordering of the nodes may lead to a smaller data structure and hence may reduce the (worst) time complexity. Still, an optimal re-ordering leading to a canonical form is known as an NP-hard problem. While a poset reordering was suggested in [111], complicated heuristics have been devised in the literature on the subject to either avoid the exponential growth of binary diagrams or of balanced R-trees. In particular, the binary tree can be organised in regards to the frequency of attributes and the variable dependencies (the analysis being an $O(n^2)$ process). In such a case, frequent attributes are positioned near the root. The R-tree can be balanced by splitting an overflowing node during the joining process. Both a linear and an enhanced quadratic method in M and in the number of dimensions have been proposed. Going one step further, the filters that overlap can be merged together [111, 41] so as to reduce the depth and width of the data structure.

5.4 Enhanced Forwarding

Forwarding a notification or a (un)subscription is a common and resource-consuming process for a router as it entails the two following tasks:

- Searching for the subscriptions that (i) match the incoming notification, or (ii) cover the incoming (un)subscription. As seen in Section 5.3.2 the matching of the notification and the establishment of the coverage both imply going through the router's subscriptions and are organised in several ways (list [112, 125], R- or B$^+$-tree [24] or binary decision tree [32] and poset [34]). With a list, the search consists of reviewing the subscriptions one by one ; with the remaining structures, the router traverses the structure, following the children that match/cover. The sequential reviewing and the traversal of the structure both involve an elementary operation that consists in either matching the notification attribute against a subscription filter or establishing the covering relationship between two filters.

- The two elementary operations are relatively complex. As an illustration, matching the attribute "String priority = 3" against the filter "Integer priority > 2" requires verifying that both have (i) the same type (String), (ii) the same name (priority) and (iii) that the value taken by the attribute (3) is indeed > 2. These elementary operations (*i.e.*, matching and establishment of a covering relationship) are repeated until any subscription that matches/covers is discovered.

It is worth stressing that each router traversed by a notification/(un)subscription often unnecessarily looks for the same subscription(s) and thereby performs a duplicate search. In order to avoid any unnecessary duplication of the search while making it easier to search when necessary our approach is threefold:

- We include in the notifications (resp. (un)subscriptions to be conveyed) the subscription filters that need to be applied to filter the notification (resp. cover the incoming (un)subscriptions). Based on these embedded filters, any intermediate router gains fast access to the subscription(s) that match(es)/cover(s) the incoming subscriptions/notifications without searching for it/them.

- We summarise/approximate the subscription filters using a Bloom filter, which is useful for several purposes (§ 5.4). First, handling a summary (*i.e.*, a Bloom filter) – in lieu of a full subscription filter – speeds up the filtering of event notifications and the establishment of a covering relationship between subscriptions. Second, rather than including in the notification/(un)subscription all the subscription filters that match/cover the notification/(un)subscription, a router adds a compact set made of the related Bloom filters. Third, Bloom filters can be easily aggregated. This means that a set of Bloom filters that are added to the notification/(un)subscription can be easily aggregated into a single and comparatively smaller Bloom filter. Based on the set of Bloom filters (or the aggregated Bloom filter) a router forwards the notification/(un)subscription to the next router without searching for the subscription(s) that match(es)/cover(s) the incoming notification/(un)subscription (§ 5.4.1).

- We index subscriptions (§ 5.4.2) so as to enhance the search for a matching/covering subscription; the index is similar to a book index that allows anyone to quickly find the chapter of interest, without sequentially going through the book. In our case, the index corresponds to a hash table, which allows us to easily find the subscriptions covering the incoming notifications/(un)subscriptions.

Subscription and notification summarising

In order to accelerate matching notifications against subscription filters, we propose to represent each filter (and also each subscription) in a compact form by using Bloom filters. A short introduction on Bloom filters is provided in the following section.

The Bloom Filter in a nutshell

Bloom filters are commonly used first to represent a set in a compact manner and subsequently to test the membership of an element. A Bloom filter [25] is a bit vector of size m, denoted $B = b(1), \cdots, b(m)$, which is initialised to 0 (i.e., $\forall i \in [1, m]$, $b(i) = 0$).
Updating a Bloom filter - To add an element e to the Bloom filter, k hash functions $h_1, \cdots, h_k$ are used. Each hash function is applied to the element and the k bits at the positions designated by $h_1(e)\ mod(m), \cdots, h_k(e)\ mod(m)$ are set to 1. Assuming that the hash time is $O(1)$ then the time spent adding an element is $O(k)$.
Testing the membership of an element - Testing whether an e element belongs to the set encoded in a Bloom filter is very similar to adding an element. The e is hashed, which gives the values $h_1(e)\ mod(m), \cdots, h_k(e)\ mod(m)$. In a Bloom filter, if a bit at said positions is 0 then the element is not stored in the Bloom filter. Otherwise (if none of the bits are at 0), the element is considered to be contained in the Bloom filter, knowing that a false positive may occur but no false negative can occur. The time required to test the membership of an element is $O(k)$.
Delete an element - An element cannot be removed from a Bloom filter simply by resetting the k bits to 0. To solve this problem, counting Boom filters have been introduced [60]. The idea is to replace the bit vector forming the Bloom filter with a u bit counter vector. A counting Bloom filter corresponds to a vector of m counters. Each time an element is added, the counters at $h_1(e)\ mod(m), \cdots, h_k(e)\text{mod(m)}mod(m)$ are incremented. Thus, an e element can be deleted by decreasing the k counters located at the positions $h_1(e)\ mod(m), \cdots, h_k(e)\ mod(m)$. A counting Bloom filter supports deletion of data in $O(k)$. However, the use of counters implies an increased use of storage space and the possibility of false negatives.

Encoding of subscription filters as Bloom filter

A subscription is composed of a set of p filters. Each subscription filter, denoted f_j (with $1 \leq j \leq p$), composing the subscription is hashed[4] using k hash functions and stored in a Bloom filter.

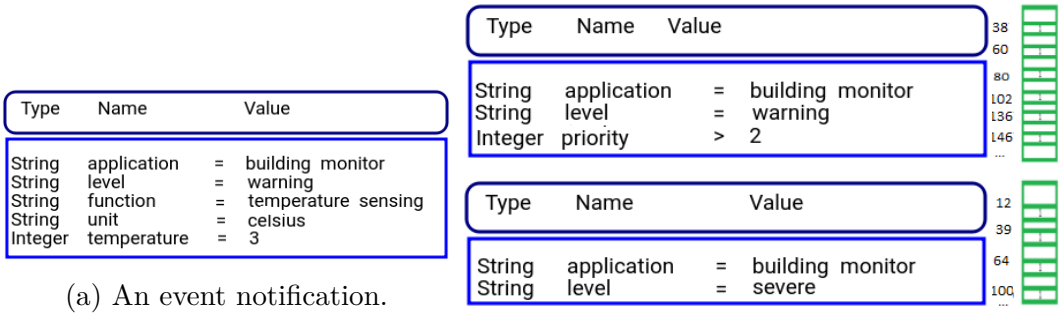
In order to limit their space footprint, there are two options for representing the Bloom filter depending on the number of elements (i.e., filters) to be stored: as a bit vector (as is the case with traditional Bloom filters) if the number of elements is low or as a list of hashes. Since a set of Bloom filters can be easily aggregated, a subscription can therefore be summarised into a single Bloom filter (Figure 5.6b). In order to summarise a set of filters, denoted $f_1, \cdots, f_j, \cdots f_p$, into a Bloom filter b_f, we apply a bitwise OR on the corresponding Bloom filters: $b_f = b_{f_1}\ OR\ \cdots\ OR\ b_{f_j}\ OR \cdots\ OR\ f_p$. If the Bloom filter is encoded as a list of hashes, the resulting Bloom filter is encoded as: $b_f = b_{f_1}, \cdots, b_{f_p}$.

Bloom-filter based repository

Each router keeps the received subscriptions in a dedicated repository, which may be organised as:

- A list of subscriptions in which each subscription is stored along with the corresponding Bloom filter that summarises the subscription. Assuming that n_f filters compose a sub-

4. Each field composing the filter can be hashed independently or more simply the entire filter can be hashed.



(a) An event notification.

(b) Two subscription with their related Bloom filters.

Figure 5.4 – An event notification matches the first subscription and does not match the second subscription

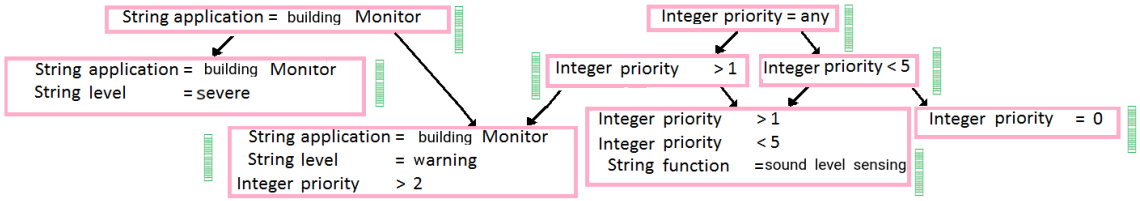


Figure 5.5 – **Subscriptions repository organised as a poset**. In the poset, each node refers to a set of filters stored along with the related Bloom filter, which is encoded as a bit vector.

scription, the space overhead associated with a Bloom filter that is encoded as a vector of bits (or a list of hashes) is $-\frac{k\ n_f}{ln(1-\rho^+)}\ k\ n_f$.

- A tree or a poset (Figure 5.5): each node of the tree/poset corresponds to one or several filter(s) that are also summarised into a single Bloom filter. Assuming that n_f filters are present, the space overhead with a Bloom filter for each node is $-\frac{k\ n_f}{ln(1-\rho^+)}$ for a Bloom filter encoded as a vector of bits ($k\ n_f$ for list of hashes).

Based on its subscription repository enriched with Bloom filters, routers forward the received notifications/(un)subscriptions to the interested recipients.

5.4.1 Preventing Intermediate Routers From Performing Duplicate Searches

Our goal is to prevent any router through which a notification/(un)subscription passes from individually performing the same treatment. The design rationale is that the first router that receives a notification/(un)subscription searches for the covering subscriptions and indicates in the outgoing notification/(un)subscription the filters that should be applied to the notification/(un)subscription. In order to illustrate this process we consider the scenario of Figure 5.2b: a device I emits the event notification that is forwarded by the routers C, A and B to the two subscribers G and H. The search proceeds as follows:

1. The first router that receives the notification – C in our example – searches for the subscription(s) that cover(s) the incoming notification/subscription and determines to which neighbouring router(s) propagate the notification/subscription.

2. Then the first router adds to the outgoing notification any indications about the location of the subscription (*i.e.*, how to access the subscription), which allows the next routers to quickly find the subscription(s) without searching for the subscription(s). The type of indication varies depending on the underlying structure adopted to store the subscriptions. With a list of subscriptions the indication consists of the ordered list of subscriptions that match (or resp. cover) the incoming notification (resp. subscription). Note that this ordered list of filters is expressed as a compact set of Bloom filters. In the case of a binary-decision-/R-tree or poset, the path(s) to follow to get the matching subscription(s) must be added. This path materialises by an ordered list of filters that are aggregated into a Bloom filter.

3. The subsequent routers (*i.e.*, A and B) rely on the provided indications (*i.e.*, the list of ordered subscriptions/filters) to extract the related subscription(s) and to forward notification/(un)subscription to the appropriate router(s).

Overall, the above outlined approach eases (and even prevents) the search for subscriptions by intermediary routers. An intermediate router that receives a (un)subscription/ notification containing Bloom filters proceeds by extracting each Bloom filter. If the repository is organised as a list of subscriptions, the router checks if the filter equals a Bloom filter of the repository, which requires $O(k \times n_f)$ comparisons per Bloom filter. If the repository is organised as a tree or poset the router goes through the structure following the path expressed in the provided Bloom filter, which requires $O(k \times n_p)$ comparisons with a path going through n_p filters.

Overall, the representation of subscriptions as Bloom filters speeds up the matching of a notification against a subscription. Nonetheless false positives may occur. In such a case, a router assumes that an incoming subscription/notification is covered/matches (while this is not the case) and unnecessarily forwards the notification/(un)subscription. The probability of a false positive differs depending on the representation of the Bloom filter.

- *Probability of false positives with a Bloom filter corresponding to a list of hashes* - Knowing that the probable occurrence of a false positive when an filter is compared to a filter is equal to $(1 - \frac{1}{N})^k$, we can therefore infer that the probability of a false positive $p(E)$ occurring when n_f filters are compared to n_e attributes is:

$$p(E) = C_{n_f}^{n_f}(1 - \frac{1}{N})^k \tag{5.1}$$

- *Probability of false positive with a Bloom filter expressed as a vector of bits* - We will use a balls-and-urns construct to model the handling of a Bloom filter. A Bloom filter (i.e., a vector of bit) is modelled as m urns. We distinguish the repository subscriptions and incoming subscription using different colours: an incoming subscription is modelled as a set of bright balls and a subscription of the repository as set of back balls. The addition of n_f filters in a Bloom filter is equivalent to throwing $k \times n_f$ bright balls into m urns (with m corresponding to the size of the Bloom filter). We define an urn as shiny if it contains at least one shiny ball. Adding n_f filters to a Bloom filter corresponds to randomly throwing $n_f \times k$ black balls into m urns. A false positive occurs if a subscription of the repository does not correspond to an incoming subscription even thought the $n_f \times k$ black balls have been thrown into bright urns. Using conditional probabilities and applying the law of total probability we infer that the $p(E)$ probability of a false positive occurring is:

$$p(E) = \sum_{r=1}^{m} \Big[\sum_{i=1}^{m} p(E/E_i/E_r)p(E_i) \Big] p(E_r) \tag{5.2}$$

with $p(E/E_i/E_j)$ denoting the probability that r black balls are thrown in i bright urns, $p(E_i)$ corresponding to the probability that i urns are bright, and $p(E_r)$ the probability that r urns are black. For a given i and given r, we observe that $p(E/E_i/E_r)$ is:

$$p(E/E_i/E_r) = \left(\frac{i}{m}\right)^r \tag{5.3}$$

In addition, the probability $p(E_i)$ that i urns are bright corresponds to the quotient between:

- The number of surjections from a set containing $k \times n_f$ elements to a set of i elements [79]. This number is equal to $i!.St(k \times n_f, i)$, with St symbolising Stirling's number.
- The number of functions of $k \times n_f$ elements in a set of size m, which corresponds to $m^{k \times n_f}$.

So:

$$p(E_i) = \frac{1}{m^{k \times n_f}} \sum_{j=0}^{i} (-1)^j \ C_i^j \ j^{k \times n_f} \tag{5.4}$$

Likewise, $p(E_r) = \frac{1}{m^{k \times n_f}} \sum_{s=0}^{r} (-1)^s C_r^s s^{k \times n_f}$. We conclude that:

$$p(E) = \frac{1}{m^{k^2 \times n_f \times n_f}} \sum_{r=1}^{m} \sum_{s=0}^{r} (-1)^s C_r^s s^{k \times n_f} \sum_{i=1}^{m} \frac{i^r}{m^r} \sum_{j=0}^{i} (-1)^j \ C_i^j \ j^{k \times n_f} \tag{5.5}$$

Based on Formulas 5.2 and 5.5, the number of hash functions is set according to the number of filters contained in the subscription/path, so that the probability of a false positive remains bellow a given threshold.

5.4.2 Prefiltering with an Index-based Subscription Repository

The efficiency associated with the event/subscription forwarding depends of various aspects. With a repository containing a list of subscriptions, key factor is the number of subscriptions stored in the repository. With tree or poset, the depth and the width of the tree/poset are two key factors. Ususally, the root tends to have many children. Visiting all the children of the root – wich we call horizontal crossing – is laborious and very frequent. Note: if there is no subscription filter that matches, all the children of the root are visited. In order to: (i) speed up the subscription search (especially the horizontal crossing) and (ii) avoid starting a subscription search when no subscription filter match, we propose to: (i) rely on hash tables that speedup the horizontal crossing (ii) introduce a prefiltering process that avoid searching for some covering subscriptions while no subscription covers.

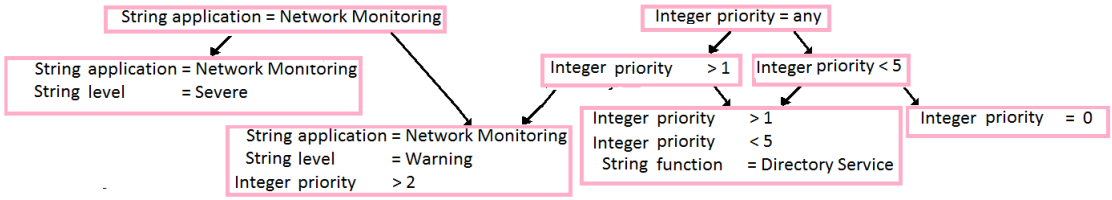
Index-based Structure

We propose to index the subscriptions, leveraging hash table. With a list of subscriptions, we propose to replace the list by a hash table. Thus, subscriptions are located with a key (as detailed in the following paragraph), without searching for the subscription. In the case of a poset or tree, the first level of nodes are also organised in a hash table. Similarly, subsequent level may also be accessed using a hash table, *i.e.*,, the children of a given node are easily

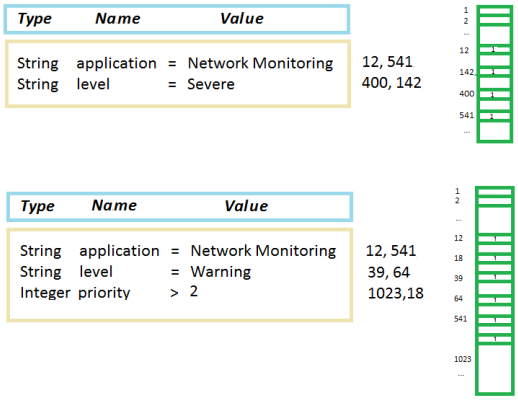
organised in hash table. Thus, upon the reception of a new subscription, it becomes easier to localise the subscription in the hash table. In the case of a (ordered) list, subscriptions are located according to a specific criterion, rather than searching for the subscription sequentially or by dichotomy (if the list is ordered). In the case of a poset (Figure 5.6a) or tree, it allows the router to quickly find the first level nodes. Similarly, subsequent levels can be accessed using a hash table.

Prefiltering

Notifications are prefiltered using a counting Bloom filter stored in each router, which encompasses all the subscriptions, *i.e.*, that aggregates all the Bloom filters stored by a router. As a result, a router avoid starting an unnecessary lookup. Moreover, several counting Bloom filter may also summarise all the filters accessible through the nodes of the poset/tree. Using these counting Bloom filters avoid starting a search from these nodes whereas there is no matching subscriptions.



(a) Poset



(b) Bloom filter associated with a filter.

Figure 5.6 – Subscriptions organised in poset and Bloom filter associated with a filter.

In the following, we briefly describe the way subscriptions are indexed and added to the router repository so that incoming notifications can be filtered.

Generating an index

We hash the names of each filter f_i constituting the subscription. A particular index designated by the key $min(h(f_i.\text{name}))$ is selected and used to store the subscription. We use the minimum value to define the position where the subscription is stored, knowing that more advanced techniques could be used to perform load balancing. In order to minimise the number of subscriptions that are stored, a subscription is only added if there is no other subscription propagated by the neighbouring router that covers this subscription. The identity of the neighbouring router is stored along with the subscription so that later notifications can be redirected to it.

Adding a subscription

Adding a subscription to the repository consists of extracting each filter and hashing the filter name so as to generate a set of keys. If the repository is organised as a list of subscriptions, then, the subscription is simply added in the cell of the hash table, which is designated by the lowest index. If the repository is organised as a tree/poset, each cell given by a key is inspected. If all the concerned cells are empty, then the subscription is simply added in the cell identified by the lowest key. Otherwise, if one cell is not empty, then there is potentially an overlapping subscription. In such a case, the covering relationship is recursively established: the covering relationship with the filter(s) composing the node is studied and the traversal continues with the matching child(ren) (if any) so as to find the proper cell wherein is ultimately added the subscription.

Looking for the subscription(s) that match(es)

Upon the reception of an event notification, the router pre-filters the notification. If the notification is not pre-filtered, then the router hashes the name of the attributes that form the subscription and inspects the location identified by the hashes so as to find the filter(s) (and possibly the subscription) that match. With a poset/tree, this process is recursively iterated so as to find the matching subscription.

5.4.3 Synthesis

We have proposed an event notification system that distinguishes itself in the use of Bloom filters that improve the lookup process, thereby introducing significant performance gain (in terms of delay and computational load). Subscription is encoded as a concise Bloom filter that is further conveyed along with the subscription (resp. notification) so as to speed up the routing by avoiding the sequential comparison of each subscription constraint (resp. each notification attribute). In addition, counting Bloom filters are used to summarise the subscriptions and avoid searching for potential subscriptions that are not existing anyway.

In the following we go one step further and focus our attention on the problem of assigning an adequate size to Bloom filters. This is critical since dealing with:

- undersized Bloom filters induces a poor accuracy (i.e., a high false positive rate with Bloom filters and false negative rate only for the counting Bloom filters).
- oversized Bloom Filters lead to a waste of bandwidth/storage usage with no gain in terms of accuracy.

Thus, the Bloom filter size should be modulated (i.e., increased or decreased) when the number of items stored in a Bloom filter substantially varies.

To cope with an undersized Bloom filter, a straightforward solution lies in adding another Bloom filter in which new items are inserted. To deal with oversized Bloom filter, a naive (but widely practiced) approach lies in recomputing another Bloom filter with the most suitable size. This is impractical because it involves a significant computation cost since all the elements must be added to the new Bloom filter.

In the following, we continue our analysis of our event notification system by focusing our attention on the compression of Bloom filter, which is a critical factor since an oversized Bloom Filter results in significant bandwidth/storage usage with no gain in accuracy. To this end we introduce a lossy compression algorithm, which consists in folding the Bloom filter, so as

to require less space for storage or transmission. Note that this approach can be applied to counting Bloom filter. The proposed compression takes advantage of the fact that Bloom filters contain more information than necessary.

5.5 Folding Bloom Filters

Folding a Bloom filter is beneficial in the two following circumstances:

1. the Bloom filter stored in memory should be folded because it was initially over-sized. This situation frequently occurs because it is difficult to find a suitable size for a Bloom filter since one must estimate in advance the amount of data that will be stored. Thus, a conservative approach based on over-sizing Bloom filters, which is often adopted.

2. A large Bloom filter needs to be transmitted and must first be folded so as to keep bandwidth usage and communication delay to a minimum.

In order to deal with both cases, we introduce a lossy compression algorithm, which performs the folding of the Bloom filter and hence optimises the bandwidth/storage usage by reducing the Bloom filter's size.

In the following, we exemplify the folding of a Bloom filter with the following Bloom filter of 20 bits: 10010 11111 11101 11000. The halving of the Bloom filter consists of applying a OR on the two portions of the Bloom filter: 10010 11111 OR 11101 11000. The resulting Bloom filter of size 10 is 10000 11000. With a counting Bloom filter, the folding is done by adding the corresponding counters, using the maximum in case of overflow. It is worst-emphasising that with a Bloom filter of size 20, there exists several possible folding. Instead, a Bloom filter containing 5 bits/counters cannot be folded[5]. Let us understand the reason why by considering a Bloom filter of size 20. A Bloom filter of size 20 can be folded in 7 ways because there are 7 ways of factorising the original size of the Bloom filter: $20 = 2.10 = 2.2.5 = 2.5.2 = 4.5 = 5.4 = 5.2.2 = 10.2$. Remark that 5 and 2 are primes while 20, 10 and 4 are some composite numbers, *i.e.*, they can be expressed as a product of primes. Their composite nature implies that a Bloom filter with a size of 20, 10 or 4 can be folded while a Bloom filter with a size equal to 5, cannot be folded because 5 is a prime. Overall, selecting a proper size for the Bloom filter is critical so as to guarantee that many different folding may be applied later. Then, among the possible choice(s), our goal is then to determine which is the best folding to apply at run time. In the following, we explore to which extend the folding may be achieved and we break the two related optimization problems into the off-line sizing of the Bloom filters and their on-line folding:

- *Off-line sizing of the Bloom filters* (§ 5.5.1) - The sizing of a Bloom filter depends on the number of elements that are expected to be added and the required false positive rate. We will show that the ability to fold a Bloom filter is related to the size of the Bloom filter. We will investigate this relationship by introducing the mathematical background on number theory as well as providing a mathematical and practical formulation for finding the optimal Bloom filter size. Although we will show that this size should be ideally a *highly composite number* (i.e., a positive number that has a larger number of divisors than any number smaller than itself), practical considerations will lead us to formulate the problem of finding a Bloom filter size as equivalent to finding a *y-smooth-number*, i.e., a positive number whose prime factors are little primes (i.e., $\leq y$). Once the size is set, the filter is created and may be compressed whenever needed by folding it.

5. We purposely exclude a bloom filter of size 1 that only reflects that the encoded set is empty or not empty

- *On-line folding* can be modulated (§ 5.5.2) based on the amount of data actually stored in the Bloom filter and the admissible false positive rate. The folding (§ 5.5.3) can be viewed as a simple form of lossy compression that reduces the size of the Bloom filter and hence requires less space for storage or transmission. The lossy nature of the compression implies that the original Bloom filter cannot be reconstructed from the folded Bloom filter.

5.5.1 Finding y-Smooth Numbers

Number theory comes into play here because the determining the precise sizing of Bloom filters is critical to our approach. So before moving on with the planning of the Bloom filter size, let us first introduce some basic notions and background on number theory and more specifically on smooth numbers.

Digression on Number Theory

Any number $m \in \mathbb{N}^*$ can be expressed in an unique way as a product of primes:

$$m = \prod_{j=1}^{\infty} (p_j)^{\gamma_j} = \prod_{j=1}^{a} (p_j)^{\gamma_j} \tag{5.6}$$

with $p_j \in \mathbb{N}$, $(i, \gamma_j) \in \mathbb{N}^2$, $a = max\{i \in \mathbb{N} \,|\, \gamma_i > 0\}$ and $\forall i < j, p_i < p_j$. Contrary to a prime number, a *composite number* has divisor(s) apart from itself and one. Keeping in mind that we aim at finding numbers that have a high number of divisors, we naturally exclude primes and focus our attention on highly composite numbers. A *highly composite number* (resp. a *largely composite number*) [129] is a number that has a larger number of divisors (resp. a larger or equal number of divisors) than any number less than itself and is herein the best (resp. good) candidate. Unfortunately, even though largely composite numbers are more crowded than highly composite numbers, both remain rare. To illustrate: 34 highly composite numbers and 87 largely composite numbers are $\leq$ 332640; example values are provided in Table 5.1. More formally [116], the number of highly composite and largely composite numbers less than x, denoted $\Psi(x)$, is subject to $\Psi(x) << ln(x)^b$ with b being constant and respectively to $exp(log^c(x)) \leq \Psi(x) \leq exp(log^d(x))$ with c and d constant, for any large x. Studying the decomposition of the largely and highly composite numbers given in Table 5.2, we can hypothesise that a number suitable for our purposes would be composed of little primes, because this reflects the ability to fold a Bloom filter and thus to smoothly reduce the Bloom filter size. This suggests considering using a *smooth number*, which is a number with only small prime factors [100]. Technically, a positive integer is said to be y-smooth if it holds no prime factor exceeding y (*i.e.*, any constituting prime factors $\leqslant y$).These numbers are fairly numerous. Theorem 5.5.1, whose detailed proof can be found in [78], gives a sense of the density of the smooth numbers that belong to an interval [1, x] by showing that the proportion of y-smooth numbers up to x that have only small primes (*i.e.*, $y \leq x^{\frac{1}{u}}$) tends to a non-zero limit which is denoted $\rho(u)$.

Theorem 5.5.1 *Let $S(x, y)$ be the set of y-smooth up to x and $\Psi(x, y)$ denote its cardinality. Given $y = x^{\frac{1}{u}}$ ($\forall u \geq 1$), this number tends to a non-zero limit as $x \to \infty$. This limit is denoted $\rho(u)$ and the cardinality satisfies [45]:*

$$\Psi(x, y) \backsim x\, \rho(u) \;\; as \; x \to \infty \tag{5.7}$$

with ρ defined as:

- $\rho(u) = \frac{1}{u}\int_{u=-1}^{u} \rho(t)\ dt$, $\forall u > 1$, and,
- $\rho(u) = 1$ for the initial condition $0 \leq u \leq 1$.

A closed form of ρ to express $\Psi(x, y)$ has been established in [72]:

$$\Psi(x,y) = \frac{1}{\pi(y)} \prod_{p \leq y} \left(\frac{log\, x}{log\, p}\right) \left(1 + O\left(\frac{y^2}{log\, x\, log\, y}\right)\right) \tag{5.8}$$

Although this theorem gives the global distribution of y-smooth numbers over a range $[0, x)$ - the values of the function ρ used in Expression 5.7 are provided in [72] (page 288) - the density of the smooth numbers is represented by the average gap in the sequence of two smooth numbers, denoted n_i and n_{i+1} with $n_i < n_{i+1}$, is about:

$$\approx \frac{n_i}{(log\, n_i)^{\pi(y)}} \ \ with\ \ c_2(y) \leq \pi(y) \leq c_1(y) \tag{5.9}$$

Based on Equations 5.8-5.9, we observe that y-smooth numbers are quite numerous. This implies that they constitute a good solution for our folding problem.

∗1	∗2	∗3	∗4	∗6	8	10	∗12	18
20	∗24	30	∗36	∗48	∗60	72	84	90
96	108	∗120	168	∗180	∗240	336	∗360	420
480	504	540	600	630	660	672	∗720	∗840
1080	∗1260	1440	∗1680	2160	2520	3360	3780	3960
4200	4320	4620	4680	∗5040	∗7560	9240	∗10080	12600

Table 5.1 – Largely Composite Numbers (extracted from the annotations provided as part as [129] from the table handwritten by S. Ramanujan). Highly composite numbers are identified by *

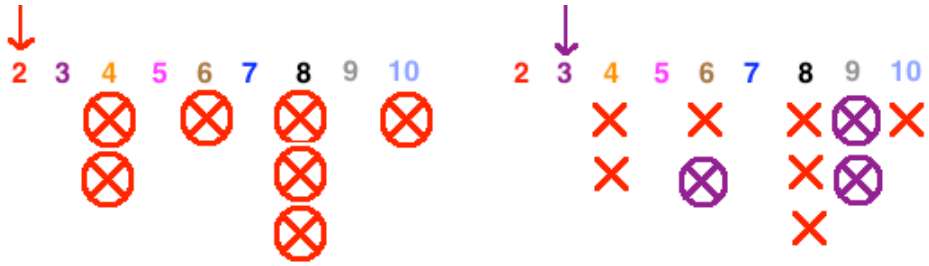
$*2$	3	$*4 = 2^2$	$6 = 2 * 3$	$8 = 2^3$	10
$*12 = 2^2.3$	$18 = 2.3^2$	$20 = 2^2.5$	$24 = 2^3.3$	$30 = 2.3.5$	$36 = 2^2.3$
$48 = 2^4.3$	$*60 = 2^2.3.5$	$72 = 2^3.3^2$	$84 = 2^2.3.7$	$90 = 2.3^2.5$	$96 = 2^5.3$
$108 = 2^2.3^3$	$*120 = 2^3.3.5$	$168 = 2^3.7$	$180 = 2^2.3.7$	$240 = 2^4.3.5$	$336 = 2^4.3.7$

Table 5.2 – Decomposition of Highly and Largely Composite Numbers

Based on a Bloom filter size, denoted x, which is provided as input by the application depending on its need, the objective is to find a y-smooth, denoted m, that is near the given size x and that offers a large number of possible folding. In practice, we isolate a short interval $[x, x + z]$ near x and find a smooth number in this interval. Note that z should be selected based on Expression 5.9 so that the interval $[x, x + z]$ contains at least one smooth number. Then, all the numbers $x, x + 1, ...x + z$ should be enumerated and the divisibility with small primes (*i.e.*, with all the primes $\leq y$) tested for each. Assuming that x is fairly large, y relatively very small and z sufficient to find at least one smooth number, excessive calculation

(a) Cross multiples of 2. (b) Cross multiples of 3.

Figure 5.7 – **Find primes up to z=10**. Primes are 2, 3, 5, 7 because these numbers are not crossed.



(a) Cross multiples of 2, 2^2 and 2^3. (b) Cross multiples of 3, , 3^2.

Figure 5.8 – **Find the y-smooth numbers up to z=10 with y=3**. The y-smooth numbers are 4, 6, 8, 9 because they hold at least two crosses.

would still be required given that $O(\sqrt{x})$ trials are expected to factorise each x. However, the cost related to the factorisation of consecutive integers $x, \ldots x+z$ can be alleviated by relying on the well-known sieve of Eratosthenes. Although we may consider more recent algorithms (*e.g.*, the elliptic curve factoring [95]), the sieve of Eratosthenes is sufficiently fast for the Bloom filters sizes that are needed in practice.

To apply the sieve, one first writes down all the numbers from 1 to z and crosses out all the multiples of 2, 3, 5 and so on, until all multiple of primes not exceeding $\sqrt{z}$ have been crossed out. The remaining numbers are the primes.

Similarly, finding the y-smooth numbers that are $\leq x$ can be done (as pointed out in [128]) by crossing out all the numbers that are powers of a prime $\leq y$ (Figure 5.8). One then counts the number of crossed-out powers for each number. If that count is sufficient, the number is a smooth number.

We will need to adopt a slightly modified formulation of the problem to generate y-smooth numbers given that we are interested in finding the y-smooth numbers that pertain to the interval $[x, x+z]$ rather than to the interval $[1, z]$. This interval encompasses the potential candidates among which the size of the Bloom filter can be selected.

For this purpose we proceed as follows (see Algorithm 3 for the details):

1. *Initialisation:*

- We start with an array of bits w of length z and initialise each array location to 0 (lines 1-2).
- We assume that the primes that are $\leq y$ (excluding 1) are known ; these primes are easily generated given that y is by construction small (*i.e.*, $y \ll x$). Let us denote by $\mathcal{P}_y$ this

predetermined set and by $\pi(y)$ the number of primes in this set.

2. Consider successively all the primes $p_i \leq y$, and for each perform the following actions:

- Find the starting point, *i.e.*, the smallest number in the interval, which can be divided by p_i and cross-out all the multiple of p_i until all are crossed (line 12), *i.e.*, cross $x+s$, $x+s+p_i$, $x+s+2\ p_i$, $x+s+3\ p^j$ ..., until all multiples of p_i within $[x, x+z]$ are crossed. Note that such a crossing-out has to be performed not only for $p_i = p_i^1$ but also (whenever possible) for p_i^2 (double crossing-out), p_i^3 (triple crossing-out) etc, which leads to the following step:

- Perform the same action (whenever possible) for p_i^2, p_i^3, p_i^4... until all the crossing-out ends. More precisely, this requires shifting the starting point to $x+s+p_i^j$ and crossing out $x+s+p_i^j$, $x+s+2\ p_i^j$, $x+s+3\ p_i^j$, $x+s+4\ p_i^j$ ..., until all multiples of p_i^j within $x, x+z$ are crossed out (line 12).

3. Finally, we attempt to enumerate the y-smooth numbers, *i.e.*, all the numbers m of [x, x+z] that hold a sufficient number of crosses.

Require: $x \in \mathbb{N}^*, y \in \mathbb{N}^*, z \in \mathbb{N}^*, z > x, \mathcal{P}_y$
1: **for** $i = 0$ to z **do**
2: w[i] = 0 {initialise the array w used to record the sum of the primes powers}
3: **end for**
4: **for** $p_i \in \mathcal{P}_y$ **do**
5: $j \leftarrow 1$
6: $start_{p_i^j} \leftarrow \left(p_i - \left(\frac{x-p_i}{2}\right) mod\ p_i\right) mod\ p_i$ {find the starting point so that $p_i^j | x+i$}
7: **while** $start_{p_i^j} \leq z$ **do**
8: **for** $k \leftarrow 0$ *to* $\lfloor \frac{z - start_{p_i^j}}{p_i^j} \rfloor$ **do**
9: $w[i + k \cdot p_i^j] \leftarrow log(p_i)$ {cross the multiple $k.p_i^j$}
10: **end for**
11: $start_{p_i^j} \leftarrow start_{p_i^j}.p_i$
12: $j \leftarrow j + 1$
13: **end while**
14: **end for**
15: **if** $w[i] \geq log(x)$ **then**
16: $SmoothNumbers = SmoothNumbers \cap x + i$ {$x+i$ is a y-smooth number}
17: **end if**

Algorithm 3: Looking for Smooth Numbers

Complexity analysis - Let us investigate the time and space complexity of the algorithm.

- *Initialisation:* the initialisation of an array whose length is z, requires $\sim z$ steps and, according to the Prime Number Theorem, the space allocated to store the primes is $O(\pi(y)) \approx \frac{y}{ln(y)}$.

- *Crossing:* the time devoted to multiple crossings-out splits into the research of the starting point $start_{p_i^j}$ and then the crossing operation starting from $start_{p_i^j}$. For all p_i the time for finding the starting point $start_{p_i}^j$ is $\approx \pi(y) \approx \frac{y}{log(y)}$. In the following, we first consider the

time devoted to the first crossing (step 2.a) and then generalise our analysis to the j^th crossing (step 2.b), with $j \geq 1$. The first crossing lies in enumerating all the multiples of p_i, with $p_i \leq y$. This operation is the one achieved by the sieve of Eratosthenes implemented in the usual way [117], the execution time being $\Theta(z\ log\ log\ y)$. More precisely, $\frac{z}{p_i}$ crossing operations are performed for a given p_i. Assuming that $y \leq z$, we henceforth obtain that for all p_i there are $\sum_{1 \leq p_i \leq y} \frac{z}{p_i} \approx z\ log\ log\ y$ crosses.

The general case consists of crossing all multiples of p_i^j, with $j \geq 1$. For a given prime p_i subject to $p_i^j \leq z$, the jth operation consists of crossing the multiples of p_i^j, leading to $\frac{z}{p_i^j}$ crosses. All p_i subject to $p_i^j \leq z$ hence involve $\sum_{1 \leq p_i \leq y} \frac{z}{p_i^j}$ crossing with $j \geq 2$. Overall, the time $\phi(x, z, y)$ needed for all crossing operations satisfies:

$\phi(x, z, y) = \sum_{p_i=1}^{y} \sum_{j=1}^{log_{p_i}(z)} \frac{z}{p_i^j}$

Let us find the upper-bound of ϕ:

$\phi(x, z, y) = \sum_{p_i=1}^{min(y,z)} \frac{z}{p_i} + \sum_{p_i=1}^{min(y,z^{1/2})} \frac{z}{p_i^2} + \sum_{p_i=1}^{min(y,z^{1/3})} \frac{z}{p_i^3} + \ldots \leq \sum_{p_i=y \leq z}^{p_i=y \leq z} \frac{z}{p_i} + \sum_{p_i=1}^{min(y,z^{1/2})} \frac{z}{p_i^2}(1 + \frac{1}{p} + \frac{1}{p^2} + \cdots + \frac{1}{p^{log_2(x+z)-2}}) = \sum_{p_i=1}^{min(y,z)} \frac{z}{p_i} + z \sum_{p_i=1}^{min(y,z^{1/2})} \frac{1-p_i^{log(x+z)-1}}{p_i(p_i-1)} \leq z\ (log\ log\ y + O(1))$

- *Enumerating y-smooth Number* - Finding all the smooth number in [x,x+z] requires an O(z) time execution.

Overall, the space required is $z + \pi(y)$ and the computation time is $\pi(y) + z\,(log\ logy + O(1))$. Note that a logarithm formulation of the problem is used in the provided algorithm so as to limit the space requirement involved in storing the crosses for any of the investigated z and for representing the $\pi(y)$ primes. It also privileges addition over multiplication by representing a composite number in its logarithmic formulation as the following sum: $log\ m = \sum_{j=1}^{a} \gamma_j\ log(p_j)$ rather than as the product $m = \prod_{j=1}^{a} p_j^{\gamma_j}$.

Among the resulting candidates, one y-smooth number, denoted m, is selected ; the smooth number should be selected to ensure that the Bloom filter is highly foldable, which yields to the following canonical factorisation of m:

$$m = \prod_{j=1}^{a} (p_j)^{\gamma_j} \quad \text{with} \quad p_i \leq y. \tag{5.10}$$

Thus the operational Bloom filter can be subsequently folded when necessary.

5.5.2 On-Line Folding Strategy

The goal is to determine the optimal folding to apply at run time, keeping in mind that the goal remains to meet the required false positives rate ρ^+ (and the required false negatives rate ρ^- for the counting Bloom filter).

We express this on-line folding strategy as the following optimisation problem. Given that m_t is the actual size of the Bloom filter, n_t the number of elements stored and k the number of hash functions, our goal is to find the minimal size of the resulting folded filter m_{t+1} that still meets the required false positive rate ρ^+ and false negative rate ρ^-, subject to m_{t+1} dividing

m_t (which is denoted $m_{t+1}|m_t$). We assume that the number of hash functions is set once, preferably to its optimal value, minimising the false positives rate and is given by $k = \frac{m_0}{n_0}\, ln(2)$, with m_0 referring to the initial size of the Bloom filter which is about $\frac{-n_0\ ln(\rho^+)}{(ln(2))^2}$ and n_0 denoting the number of items that is expected to be added to the filter. We find that the m_{t+1} that still meets the false positives rate satisfies:

$$m_{t+1} \geq \frac{-n_0\ m_t\ ln(1-(\rho^+)^{1/k})}{n_{t+1}\ f_0...f_t\ ln(2)} \tag{5.11}$$

Given that our goal is also to find m_{t+1} that divides m_t, the above expressed goal can be refined as finding all the combinations of $p_i^{\gamma_i}$ which produce m_{t+1} and selecting one combination. For instance, as suggested in the following algorithm, the minimal m_{t+1} that matches Equation 5.11 can be used.

Recall that with $m_t = \prod_{j \in S_t} p_j^{\gamma_j}$ the exhaustive enumeration of any folding is characterised by a worst case computing time that is exponential in $\sum_{j \in S_t} \gamma_j$ because the maximum number[6] of distinct products m_t obtainable is $2^{\sum_{j \in S_t} \gamma_j}$. The storage required is $\sum_{j \in S_t} \gamma_j$ in the worst case. But if some combinations of folding are equivalent (*i.e.*, have the same product) and are henceforth not computed and not stored, the number of possible folding becomes in the order of $\sum_{j=0}^{q_t-1} \binom{q_t}{j}$ with $q_t = \sum_{j \in S_t} \gamma_j$. Then if the overall set of folding which is computed for m_0 is ordered (the duplicates being pruned), the computational cost related to finding m_{t+1} can be reduced to the order of $(log(\sum_{j=0}^{q_t-1} C_q^j))$ with a worst case of $\mathrm{O}(\sum_{j \in S_0} \gamma_j)$. Once the size of the Bloom filter is fixed, the Bloom filter can be folded.

5.5.3 Folding

The folding of a Bloom filter consists in first dividing the Bloom filter into segments of size m', which are denoted $s_1, \cdots, s_{m/m'}$. Next, these segments are combined to create a folded Bloom filter which is expressed as $s_1 \oplus \cdots \oplus s_{m/m'}$, with the folding operator $\oplus$ defined as a combination corresponding to an or logic for a Bloom filter and an addition for a counting Bloom filter with. Precisely, with a Bloom filter (resp. counting Bloom Filter) the folding corresponds to applying a bitwise or on (resp. summing[7]) the segments.

The Bloom filter is expressed as follows:

$$B' = \begin{pmatrix} \oplus_{l=0}^{p_l-1} b(1 + l\, m') \\ \cdots \\ \oplus_{l=0}^{p_l-1} b(i + l\, m') \\ \cdots \\ \oplus_{l=0}^{p_l-1} b(m_{t+1} + l\, m') \end{pmatrix} \quad \text{with} \quad B = \begin{pmatrix} b(1) \\ \cdots \\ b(i) \\ \cdots \\ b(m) \end{pmatrix}$$

This folding corresponds to a lossy compression that leads to the establishment of a Bloom filter whose size is less than or equal to that of the filter before folding. Given that a Bloom filter is a vector of size m whose constituent elements are counters whose values go from $\{0 \cdots 2^u - 1\}$,

6. This number is in fact achieved for some m_t.
7. if a sum exceeds the maximum value $2^u - 1$, the counter is set to $2^u - 1$

we have proved that the folding function φ: $\{0 \cdots 2^u - 1\}^m \longrightarrow \{0 \cdots 2^u - 1\}^{m'}$ satisfies the following properties:

1. *Compression with loss* - φ maps a set of cardinality m to a set of cardinality m' such that m' divides m for any $m', m \in \mathbb{N}^*$.
   Note: a Bloom filter B_m is indexed by its size m and a folding φ_{p_l} by the folding ratio $p_l \in \mathbb{N}^*$; this ratio can be expressed as the following quotient: $p_l = m/m$.

2. *Invariance* - The folding always provides the same result and the operations corresponding to the addition of elements, the membership test and the deletion of an element in the case of a Bloom filter, are all performed on a folded Bloom filter the same as they would be on an unfolded filter. The membership test, which is denoted ω, is performed on a folded filter $\varphi_{p_l}(B_m)$ as it would be on a filter of size $m' = m/p_l$ with $\omega_k(\varphi_{p_l}(B_m)) = \omega_k((B'_m))$.

3. *Commutativity* - A sequential folding, denoted o, corresponding to two folds φ_{p_l} and φ_{p_r} is commutative: $\varphi_{p_l}\, o\, \varphi_{p_r} = \varphi_{p_r}\, o\, \varphi_{p_l}$

4. *Associativity* - sequential folding is associative: $\varphi_{p_l} o\, (\varphi_{p_r} : o\, \varphi_{p_v}) = (\varphi_{p_r}\, o\, \varphi_{p_l})\, o\, \varphi_{p_v}$

5. *Efficiency* - Any φ folding of a Bloom filter $B_m \in \{0 \cdots 2^u - 1\}^m$, $\varphi(B)$ can be done in a polynomial time.

Verifying that the folding satisfies Condition 1 is straightforward. Condition 2 mainly relates to the capacity to query the membership of an element and to add or delete this element in a folded Bloom filter. In the following we concentrate on proving that Condition 2 applies to the membership test. The proof for the addition and deletion, which can be deduced from this equation, will be left to the reader. Recall that the membership query proceeds as follows: the element e is hashed with the k hash functions and if all the k investigated bits are set to 1, e is said to be stored. Thus Condition 2 holds if, for all k, $b(h_k(e)\ mod\ m) \leq b'(h_k(e)\ mod\ m')$. Given that $b'(h_k(e)\ mod\ m') = \oplus_{l=0}^{p_l-1} b(h_k(e)\ mod\ m' + l\, m') \geq b(h_k(e)\ mod\ m)$ it follows that Condition 2 is satisfied and that the resulting probability of false positives ρ^+' on the folded (counting) Bloom filter B containing n elements, using k hash functions satisfies: ρ^+' $= (1 - (1 - \frac{1}{m'})^{kn})^k \approx (1 - e^{\frac{kn}{m'}})$. For Conditions 3 and 4, we consider a folding method that is sequentially performed.
Lemma - The sequential folding φ_{p_l} and φ_{p_r} satisfies:

$$\varphi_{p_l} o\, \varphi_{p_r} = \varphi_{p_r\ p_l} \tag{5.12}$$

Proof: Given a Bloom filter B, we can prove that:

$$\varphi_{p_l}\, o\, \varphi_{p_r}(B) = \varphi_{p_l\, p_r}(B)$$

The successive folding of B leading to B' and the subsequent folding of B' resulting in B'' can be expressed as:

$$\forall i \in [1, m''], b''(i) = \varphi_{p_l}\, o\, \varphi_{p_r}\Big(b(i)\Big) = \varphi_{p_l}\Big(b'(i)\Big)$$
$$\text{with: } \forall i \in [1, m'], b'(i) = \oplus_{r=0}^{p_r-1} b(i + r \cdot m').$$

$$\text{Thus, } \forall i \in [1, m''], b''(i) \text{ can be expressed as:}$$
$$b''(i) = \oplus_{l=0}^{p_l-1} \oplus_{r=0}^{p_r-1} b(i + l \cdot m'' + r \cdot m')$$
$$= \oplus_{l=0}^{p_l-1} \oplus_{r=0}^{p_r-1} b(i + l \cdot \frac{m}{p_l\, p_r} + r \cdot \frac{m}{p_r})$$

A one-time-composite folding of B that leads to B'' is performed as follows:

$$\forall i \in [1, m''], b''(i) = \phi_{p_r\, p_l}(b(i))$$
$$= \oplus_{j=0}^{p_r\, p_l-1} b(i + j \cdot m'')$$
$$= \oplus_{j=0}^{p_r\, p_l-1} b(i + j \cdot \frac{m}{p_r\, p_l})$$

Thus, $\varphi_{p_l}\, o\, \varphi_{p_r}(b) = \varphi_{p_r\, p_l}(b)$ given $l \in [0, p_l]$, $r \in [0, p_r]$, and, $j = l + rp_l$ $\square$.
The above lemma means that the Bloom filters resulting of a sequential folding $\phi_{p_r} o \phi_{p_l}$ is identical to a single folding $(\phi_{p_r\, p_l})$. Based on this lemma, the proof of conditions 3 and 4 can be established as:
$\varphi_{p_r}\, o\, \varphi_{p_l}(B_t) = \varphi_{p_l\, p_r}(B_t) = \varphi_{p_r\, p_l}(B_t) = \varphi_{p_l}\, o\, \varphi_{p_r}(B_t)$. Thus, $\varphi_{p_l}\, o\, \varphi_{p_r} = \varphi_{p_r}\, o\, \varphi_{p_l}$ $\square$
In addition, $\varphi_{p_l} o\, (\varphi_{p_r}\, o\, \varphi_{p_v}) = \varphi_{p_r\, p_l\, p_v} = (\varphi_{p_r}\, o\, \varphi_{p_l})\, o\, \varphi_{p_v}$
Thus, $\phi_{p_l} o\, (\varphi_{p_r}\, o\, \varphi_{p_v}) = \varphi_{p_l} o\, (\varphi_{p_r}\, o\, \varphi_{p_v})$ $\square$

The commutative and associative properties of the folding imply that the order of the folding does not matter. More generally, given the folded Bloom filter B, it is impossible to guess whether the Bloom filter went through a one-time or sequential folding; the folding pathway cannot be inspected given the resulting Bloom filter. Although the sequential folding $\phi_{p_r} o \phi_{p_l}$ and a one-time folding $\phi_{p_r\, p_l}$ provide the same result, the cost in terms of time resulting from a sequential folding is greater than the cost of a one-time folding. Indeed (as defined in Condition 5), the computation cost associated with the folding is $O(m)$: it depends on the size of B even if this cost can be reduced if the folding is parallelised. But, the cost of a sequential folding is $O\Big(m\,(1 + \frac{1}{p_l})\Big)$ with p_l defining the second folding ratio.

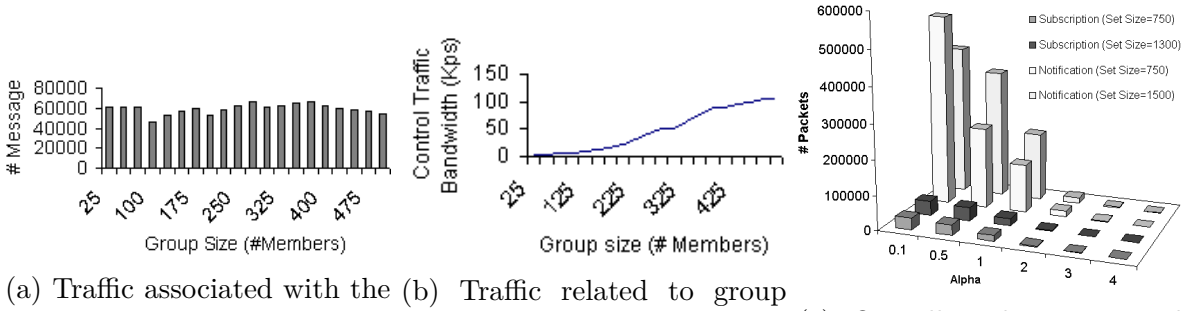
5.6 Performance Evaluation

Our next step is to assess the performance of our event notification system 5.6.1 via the folding of Bloom filters 5.6.2.

5.6.1 Event Notification System

In order to reveal the scaling properties of our event notification system, we distinguish the corresponding control traffic induced by the management of the cluster-based structure (Figures 5.9a and 5.9b) and by the dissemination of the event notifications and (un)subscriptions. For this purpose we use a constant load of 3 events/s per publisher. The distribution of subscriptions and notifications follow a zipf law[8] ; the probability of occurrence of the i^{th} subscription/notification is given by $\frac{1}{i^{\alpha}}$. We will then aggregate the results obtained from several runs over a period of 20 minutes. As illustrated in Figures 5.9a and 5.9b, the group size has a low impact on the amount of control traffic ; the bandwidth usage being indifferent to the group size, highlighting the benefits of maintaining membership information restricted to a cluster. When we consider the traffic associated with the event notification and (un)susbcription routing (Figure 5.9c), we

8. Zipf distribution has been observed by stock quote providers [172].



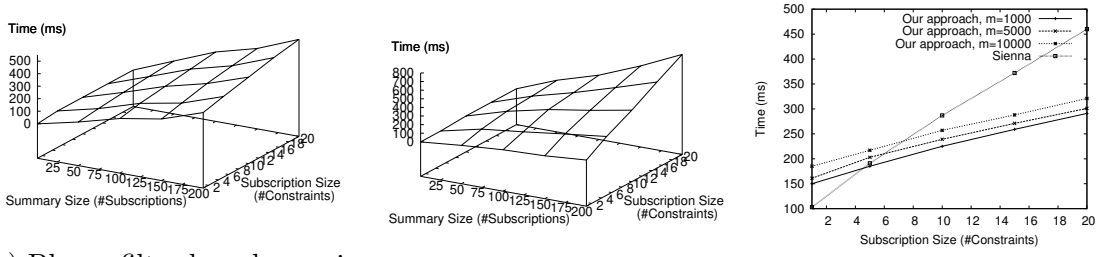
(a) Traffic associated with the whole group management.

(b) Traffic related to group management per group member.

(c) Overall susbcription and notification traffic for a group of 70 members.

Figure 5.9 – Bandwidth usage associated with our notification system.

can see that the volume of subscriptions and notifications decreases as their similarities (*i.e.*, popularity ranking given by *alpha*) increases. This reveals the combined effect of (i) leveraging the subscription covering and (ii) the aggregation and filtering capabilities made possible thanks to our cluster-based routing.



(a) Bloom-filter based repository, with k=3 and m=6000.

(b) Sienna repository.

(c) Router repository.

Figure 5.10 – Adding a subscription organised in poset and tree.

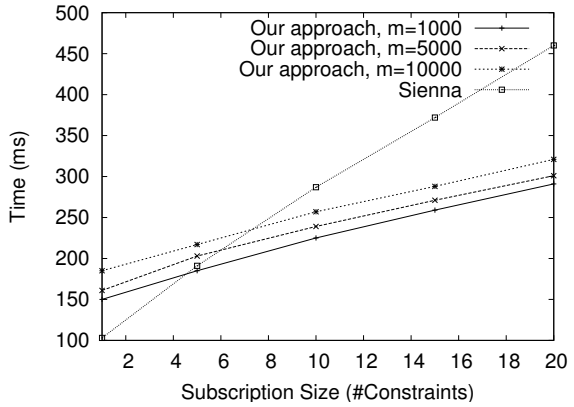
Following this we will measure the respective performances of, in term of latency, the main stages constituting the subscription and notification processes, *i.e.*, the addition of a subscription into a router repository and the filtering of an event notification. These results are then compared with the poset-based repository promoted by Siena [34], using the delay as an accurate indicator of the processing overhead incurred within each of the above stages. In order to test the impact of a complex subscriptions and notifications scanning, we further vary the subscription size (number of constraints) and notifications (number of attributes).

Subscription Insertion

We measure the latency associated with adding a subscription into a repository by considering both our approach (Figure 5.10a) and the Siena approach (Figure 5.10b), with regards to a varying number of (i) subscriptions in the repository ([1, 200]) and of (ii) subscription filters ([1,20]). Results show that Siena performs better than our approach for a number of subscription constraints inferior to 5. To illustrate: if we consider 5 constraints per subscription filter, we observe in our approach an average time ratio per subscription of 0.35 and an upper limit of 283ms and a corresponding ratio of 0.44 and maximum of 369ms with Siena. This reflects the computing cost of indexing and summarising the new subscription. However our approach

outperforms Siena when the number of constraints per subscription is greater than 5 because the indexing and summarising costs are compensated by the significant gain induced by the research covering the new subscription. Indeed, if we consider 10 and 20 constraints per subscription filter respectively, we measure a lower latency (425ms and 499ms compared to 430ms and 761ms obtained with Siena) and slower ascending rate (0.495 and 0.5925 with our approach compared to the 0.525 and 0.87 experienced with Siena). Note that this difference, in terms of ascending rate and maximum latency, increases when we consider an increased repository size (*i.e.*, number of subscriptions summarised), demonstrating that our event filtering system performs better in large-scale networks.

A similar trend is observed in Figure 5.11a. The delay incurred by our approach and Sienna increases in a linear way, our system performing better when the number of subscriptions reaches a certain threshold (from 5 up to 7 filters per subscription) whose value depends of the value of m. As expected, the latency induced by the insertion of a subscription into the repository depends of the repository size. However such impact remains limited and is easily minimised by adequately configuring our event notification system and in particular the Bloom filters' sizes (§ 5.6.2).

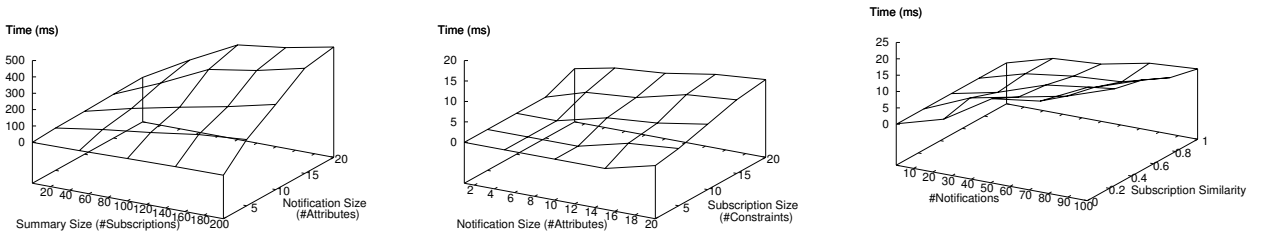


(a) Adding a subscription into a router repository.

Notification Filtering

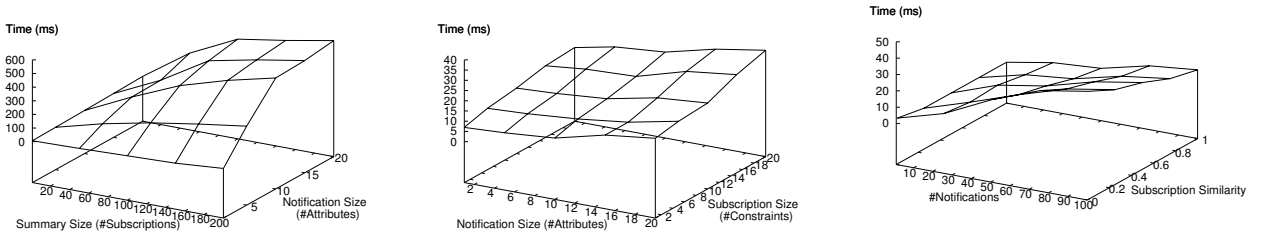
Next we compare the performances of the event notification filtering process in our solution (Figure 5.12) and Siena (Figure 5.13), considering varying repository sizes (Figures 5.12a and 5.13a), varying subscription sizes (Figures 5.12b and 5.13b) and varying notification sizes (Figures 5.12c and 5.13c). In all of the above cases we observe that the minima, maxima and ascending slope obtained by our approach are always inferior to those of Siena. If we look at Figures 5.12a and 5.13a, the difference is mostly due to the combined effect of using a summary and an index-based structure that allows the acceleration of the retrieval of the subscription(s) matching an incoming notification. The linear increase of the delay in Figures 5.12b and 5.13b is attributed to the sequential comparison of each subscription filter against each notification attribute; the lower rate of increase exhibited by our solution is due to the compact digest used to summarise the subscription.

Finally, in Figures 5.12c and 5.13c, we measure delay as a function of repository size ([1,200]) and subscription similarity (given by the ratio of covered attributes shared by the subscriptions which are stored in the router repositories). The delay decrease experienced by both our approach and Siena when considering an ascending similarity, is due to a reduced number of



(a) #constraints=10. (b) #constraints=10 . (c) #constraints=10 and #attributes=10.

Figure 5.12 – Notification Filtering with our Approach, k = 3, m = 6000.



(a) #constraints=10. (b) #constraints=10. (c) #constraints=10, #attributes=10.

Figure 5.13 – Notification Matching with Siena.

matching.

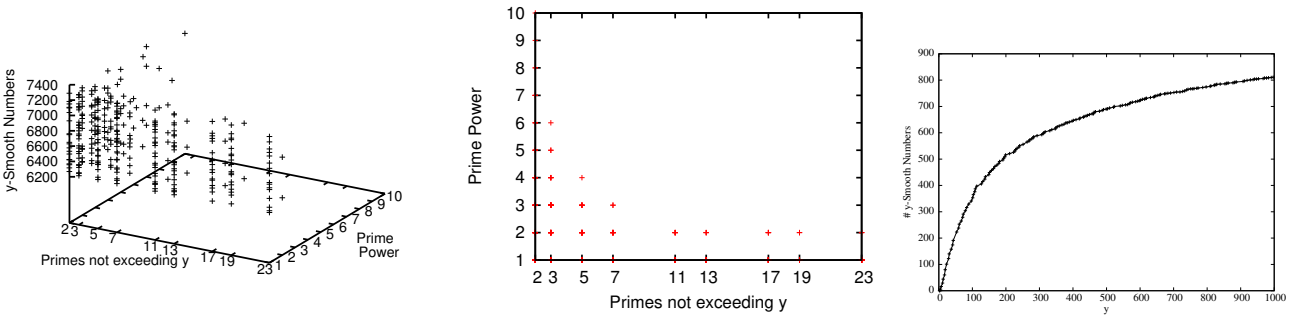
We further supplement this filtering analysis with an evaluation of the traffic generated by the event system.

5.6.2 Evaluation of the folding

Our objective is to empirically evaluate the effectiveness of the sizing (§5.6.2) and folding (§5.6.2) of Bloom filters. The results presented hereafter, are averaged over at least 100 runs and were obtained using a laptop with an Intel(R) Core(TM) 2 duo P8600 CPU of 2,40Ghz and a cache size of 2 MB.

Off-line Planning

Our primary goal is to (i) investigate the ability to select a Bloom filter size that permits many different foldings and (ii) analyse the related computing cost evaluated in terms of delay. To this end, we explore the properties of smooth numbers, including their density and their composition. Guided by our application, we limit the range of numbers that are investigated to the practically relevant cases in which the size of the Bloom filters is determined so that the probability of false positives is low. In our experiments, we first start by considering a range of false positives rates that is low and narrow (see Figure 5.14). We then reduce the range (see Figures 5.15 and 5.16) and show that there remain enough smooth numbers to pick the best Bloom filter size. In addition, we investigate to what extent this reduction affects the composite nature of the y-smooth numbers. Figure 5.14.a highlights the properties of y-smooth numbers by

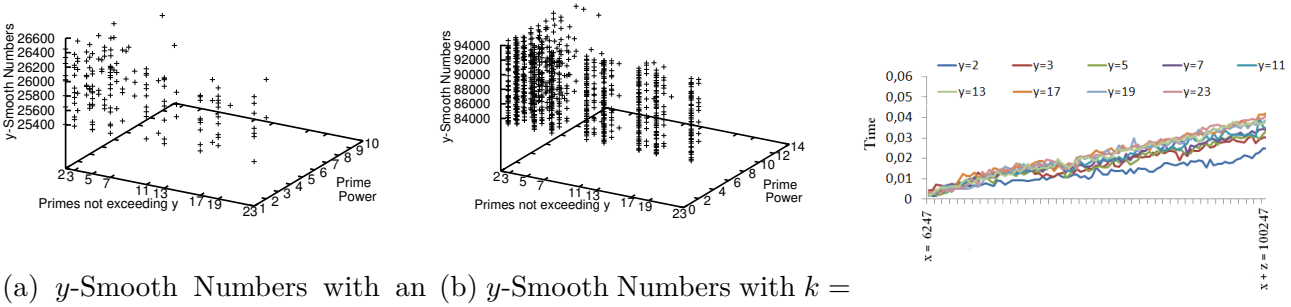


(a) y-Smooth Numbers with an optimal $k = 4$, $y = 23$.

(b) Power of the primes constituting the y-smooth numbers, $y = 23$.

(c) Amount of y-smooth numbers.

Figure 5.14 – Effectiveness of the Planning , considering n =1000, $3.10^{-2} \leq \rho^{+} \leq 5.10^{-2}$.

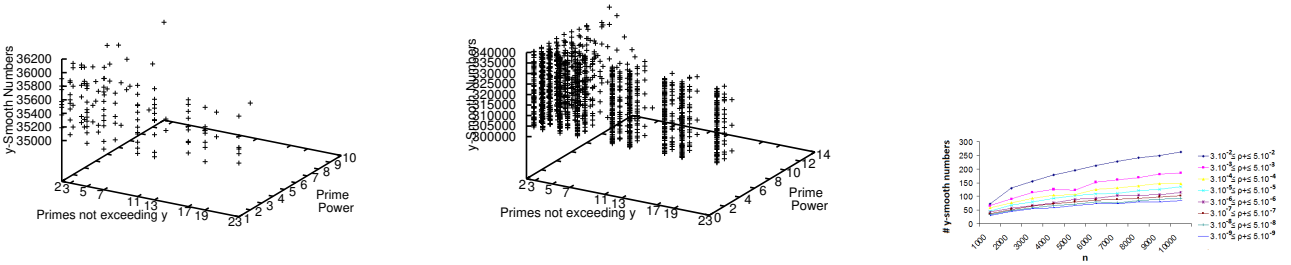


(a) y-Smooth Numbers with an optimal $k = 17$, $y = 23$.

(b) y-Smooth Numbers with $k = 4$, $y = 23$.

(c) Delay with $k = 4$.

Figure 5.15 – Scalability of the Planing, setting n =1000, 3 $10^{-6} \leq \rho^{+} \leq 5\ 10^{-6}$

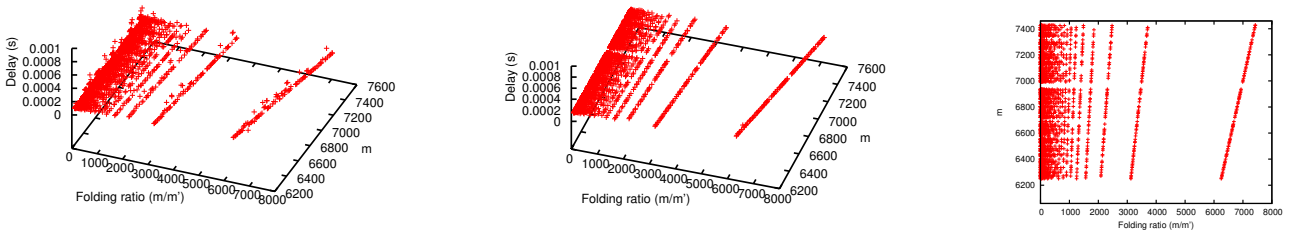


(a) y-Smooth Numbers with an optimal $k = 24$, $n = 1000$.

(b) y-Smooth Numbers with $k = 4$, $n = 1000$.

(c) Amount of y-smooth numbers.

Figure 5.16 – Scalability of the Planning, considering $3.10^{-8} \leq \rho^{+} \leq 5.10^{-8}$, $y = 23$



(a) Folding delay for a Bloom filter.

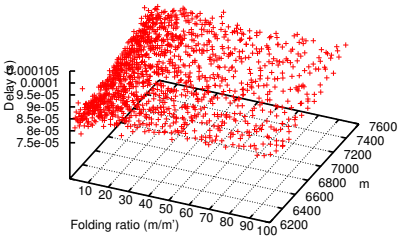
(b) Folding delay for a counting Bloom filter.

(c) Foldability of a (counting) Bloom filter.

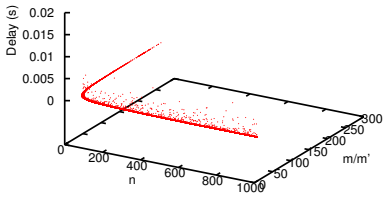
Figure 5.17 – Folding a Bloom filter, y=23, $6247 \leq m \leq 7433$.

plotting the y-smooth numbers as a function of the primes $p \leq y$ that compose them and their respective powers. The Figure also provides a quantitative support for observing the density of the smooth numbers: 11.86% of the numbers are y-smooth numbers. This is more than sufficient for the selection of a satisfactory Bloom filter size. Furthermore, the density remains stable: for instance, only 9 numbers need to be explored before finding one 23-smooth number and only 101 before finding ten 23-smooth numbers. It is also visible that the smallest primes are mostly present in the composition of the smooth numbers. They[9] are typically compensated by larger powers: this is illustrated in Figure 5.14.b where power is plotted as a function of the prime. This means that the Bloom filter can be folded many times and in a smooth fashion. In Figure 5.14.c a saturation of the amount of y-smooth numbers is observed when increasing y. At a certain point it is unnecessary to continue increasing y because the time to search for y-smooth numbers grows significantly (Figure 5.14.c) whereas the quantity of smooth numbers grows only slightly.

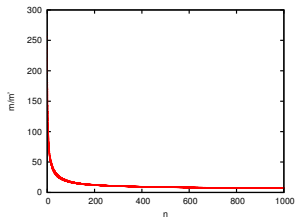
By restricting the range of the false positives rate from $\rho^+ \in [3.10^{-4}, 5.10^{-4}]$ in Fig. 5.15 to $\rho^+ \in [3.10^{-8}, 5.10^{-8}]$ in Fig. 5.16, we see that the amount of smooth numbers naturally decreases as the investigated range narrows. However the amount of smooth numbers remains significant (Fig. 5.16c): if the number of hash functions k is smaller than the optimal number of hash functions $k = 4$ (see Figures 5.15.a and 5.15.b), the amount of y-smooth numbers is comparatively much greater. This illustrates the trade-off made between time (k) and space (m): an attempt to improve the performance of one is done at the expense of the other. Nevertheless, as demonstrated in Figure 5.16.c, a sufficient amount of y-smooth numbers is still available even considering a worst case, *i.e.*, a very narrow range of false positives rates. It can also be observed that the quantity of smooth numbers is proportional to the number of elements n that are added to the Bloom filters, because the amount smooth numbers within an interval $[m_1, m_2]$ is proportional to the number of items n to be added. Indeed, $m_1 - m_2 = \frac{n}{ln(2)^2}\ ln(\frac{\rho_1^+}{\rho_2^+})$ with ρ_1^+ and ρ_2^+ denoting the false positives rate corresponding to m_1 and m_2, respectively. In the following experiments we study the folding of a range[10] of sizes for a Bloom filter, which is given by [6247,7433].



(a) Foldability of a (counting) Bloom filter, $m/m' \leq 100$.



(b) Delay for a Bloom filter.



(c) Compression ratio of a Bloom filter.

Figure 5.18 – Folding and Compression and Decompression of a Bloom filter,$6247 \leq m \leq 7433$.

On-Line Folding

In order to evaluate the performance of our lossy-compression we compare the computation cost, measured in terms of execution time, to the space economy, expressed as a compression ratio (and corresponding to the original size of the Bloom filter divided by its folded size). We also compare the cost and space economy of our lossy compression against the popular compression format Zlib[11][44] that creates a zip file. Our goal is to give the cost/gain associated with the folding relative to the Zlib compression even though the two compression methods are somehow not comparable because our folding-based lossy compression is specific to Bloom filters, while Zlib provides a lossless compression for general classes of data including *e.g.*, text. We use MurmurHash[12], a non-cryptographic hash function which is typically used for hash-based lookup. We consider the delay associated with the folding of a Bloom filter (Figure 5.17.a) and of a counting Bloom filter (Figure 5.17.b). This delay is expressed as a function of the size of the Bloom filter (m) and of the folding ratio m/m'. The Bloom filter and the counting Bloom filter both exhibit the same behaviour during folding. The easily discernible lines are related to the repetitive combinations of the same primes: the lower y, the higher the repetition. Although it is less observable in lower folding ratios (Figure 5.18.a), an in depth analysis showed us that this behaviour remains unchanged regardless of the folding ratio. Returning to Figures 5.17.a and 5.17.b, the latency, as expected increases as a function of the Bloom filter size (recall that the theoretical cost is $O(u\,m/f)$, with $u = 1$ for a pure Bloom filter) even though the quantitative cost is several times higher in average for a counting Bloom filter. This points out the price to paid for handling counters rather than bits. The arithmetic addition required to fold a counting Bloom filter is particularly costly compared to the bitwise OR used with Bloom filter. Considering the lossless compression of a Bloom filter (Figure 5.18.c) shows us that the compression ratio expressed as a function of the number of elements n in the Bloom filter decreases drastically and stabilises as the Bloom filter is populated. Meanwhile the delay related to the lossless compression remains stable (Figure 5.18.b) and lower than the folding delay (Figure 5.17), but the two delays are still comparable for a given folding ratio. Nonetheless, the ratio m/m' is greater with the folding than with the lossless compression. Specifically (see Figure 5.18.c), the highest compression ratio is obtained with an empty Bloom

9. Similarly, a high prime is compensated by either few small primes and/or smaller powers.
10. This range is the one used in Figure 5.14.
11. http://www.zlib.net/
12. https://sites.google.com/site/murmurhash/

filter (*i.e.*, a vector of entirely set to 0) whereas the lowest ratio stands for the highest[13] value of n (here, 1000). The monotonic decrease and the small compression ratio reflects the difficulty of compressing random data (*i.e.*, a Bloom filter wherein some bits are randomly set to 1) and the practical advantage of a folding-based lossy-compression.

5.7 Conclusion

The IoT is characterised by a large number of *Things* scattered around the world that need to exchange messages over the Internet. The provision of a reliable and timely means of communication over a wide area network is still a critical and problematic issue. We have addressed this issue through the introduction of an event notification system that supports a lightweight content-based filtering and that relies on a self-configuring grouping service that dynamically organises the event delivery structure. Our content-based event notification system supports a scalable group communication and novel lightweight event filtering scheme that keeps the load on routers to a minimum. Our event notification system is automatically deployed without requiring human intervention and is fault-tolerant as it adapts dynamically to any permanent or transient network failure.

It is worth stressing that the routers traversed by a notification/(un)subscription dedicate significant time and effort to search the matching subscriptions while the search is often the same for all the routers. If one router – ideally, the first router receiving the notification/(un)subscription – can indicate how to access the subscriptions, then subsequent routers do not have to repeat the same time-consuming subscription search. In order to support such an approach without congesting the network and routers, we represent the subscription as highly compact Bloom filters. In this way it is possible to effectively filter event notifications. In addition, we propose a hash-based indexing structure that allows subscriptions to be quickly retrieved, possibly based on the provided indications.

Still, finding the correct Bloom filter size is challenging, because the quantity of data must be estimated in advance to ensure that the required false positive rate is always respected. In general practice this leads to a situation in which the Bloom filter is oversized. In order to overcome this issue we propose – and further evaluate the performance of – a lossy compression system that consists of folding the Bloom filter. We show that the foldability depends on the the exact size of the Bloom filter, thereby establishing an interesting link between number theory and Bloom filters. More precisely, we propose to select a y-smooth number due to their high density and ease of production, which derives from their simple multiplicative structure. We show that selecting y-smooth numbers is advantageous since they allow a large number of possible folding combinations. We then present an algorithm based on a sieve of Eratosthenes, which is efficient with regards to the envisaged sizes of Bloom filters. Once properly sized, the Bloom filter can be folded very efficiently so that the false positive/negative rate is always met. We further analyse the ease of selecting a size for Bloom filters that allows a large number of folds and evaluate the associated costs. This led us to empirically explore the properties of y-smooth numbers, their density and composition.

Overall, Bloom filter folding is advantageous in two application scenarios. The first assumes a Bloom filter stored in memory, which needs to be folded; this is advantageous, for example, if the Bloom filter was initially over-sized. The second scenario, and the main motivation for our work, deals with the transmission of a Bloom filter, which should be folded prior to transmission

13. Although a highly populated Bloom filter would exhibit a high compression ratio, it would be subject to a unacceptable false positives rate.

in order to minimise bandwidth usage and communication delay. After appropriate sizing , the remaining problem in both scenarios is the determination of the applied folding rate in order to achieve the required false positive/negative rate. Ultimately this folding and filtering problem is a classic compromise between time (computation) and space (storage and communication): increasing the performance of one is inevitably to the detriment of the other.

Chapter 6

Conclusion

The forward-thinking academic John Culkin remarked that “we become what we behold. We shape our tools and then our tools shape us.” The truth behind this observation written in 1967, has perhaps never been more obvious than it is today. We are witnessing a unique stage in mankind’s technological history and just as technology evolves – and continues to evolve – so do we too change with it. From this perspective, the IoT has greatly increased our interconnectedness and thereby our understanding of the physical world around us. The shift towards widespread use mobile, consumer-centric smartphones is making our immersion with computing into a permanent, and somehow dependant, state by permeating practically all aspects of our daily lives. The real and the virtual have remained relatively dissociated since the dawn of information technology, but are nowadays becoming increasingly intertwined.

Shaped by the Internet of Things, my own research journey is anchored in these last decades, anticipated by Mark Weiser (1988), which has predicted a world in which "technology recedes into the background of our lives." Since then, the IoT ecosystem has been successfully applied to enhance our interconnection with physical spaces. Due to its ubiquity, the IoT might well aspire to become the *lingua franca* of a newly emerging generation of people long immersed in an IoT-landscape. However the path leading towards ubiquitous computing raises some challenges and significant research efforts remain to be invested as tribute to the discrepancy in the quality of data and services contributed by the Things.

6.1 Summary of Contributions

In an increasingly interconnected IoT where Things freely exchange data and offer services that are accessible through open and poorly secured interfaces, the IoT raises some concerns related to reliability, faithfulness and accuracy. In the last mile, IoT networks are predominately driven by – and built upon – wireless networks (Near Field Communication or slightly longer WiFi or Bluetooth-enabled technologies) that: (i) allow greater access to outsiders that exploit vulnerabilities and may cause significant damage, and (ii) make it easier for insiders to avoid identification. Monitoring the behaviour of things and the network is necessary in order to spot intrusions, coming from either inside or outside the network. Intrusions manifest in the forms of: (i) attack behaviour which differs to that of a normal user, or as (ii) a sequence of actions undertaken by the attacker that characterises the attack pattern. We have covered both forms of attacks and explored signature-based and anomaly detection, focusing on the communication and computing infrastructures that span different scales from the MANETs composed of personal laptops equipping mobile users to RFIDs tags that are highly miniaturised. Unlike mobile laptops that may undertake advanced analytics, RFID tags are passive and unintelligent; these features imply their inability to monitor and understand their environment. In each case, we have considered the nature of the monitored system (network/computer/object and supplied service) and tailored the detection system accordingly. To secure MANETs, we have introduced a distributed signature-based Intrusion Detection System (IDS) intended to monitor the attacks against the OLSR routing protocol. We noted that the device has a limited observability and

that the partial network view, which is gained thanks to other peers, is questionable in terms of reliability. To address this issue we have devised an entropy-based trust system that deals with uncertainties and incompleteness involved in trust relationships, leveraging a statistical gathering approach that gracefully and incrementally gathers observations without sacrificing detection accuracy. As a complement we have explored anomaly detection to identify any deviations from normal behaviour. Relying on Kohonen maps, which constitute an efficient way for automatically categorising, our IDS further compares the tagged behaviour against the normal user's behaviour as expressed in the user's profile.

A lack of security is not the only flaw in the IoT. The loss of data quality is a highly debated issue within the academic and practitioner communities who often deal with the unintentional corruption of sensed data. A conflict between end-users' expectations over the data and the reliability of data gleaned from low-cost or DIY sensing devices is often present. Sensor calibration in the field is essential to overcoming this issue as it: (i) ensures a proper operation of the sensing devices, and (ii) deals with varying external conditions (*e.g.*, solar radiation, freezing) and other factors (*e.g.*, activity of the end user) that affect the measurements over time. Sensor calibration has received surprisingly little attention from researchers. We have thus proposed a holistic approach to reduce the cost associated with calibration in smart spaces at scale, from the fixed Things that make up the IoT infrastructure to the mobile Things that people carry. We have developed a plan for the calibration of a large number of often inaccurate sensors in a smart space using high-integrity reference sensors that are mobile, so as to maintain an adequate sensing accuracy while minimising the required effort from the mobile calibrators. We have in turn generalised the automated calibration to the multi-party case to sustain a macro-calibration. In particular we have conceived a distributed, opportunistic macro-calibration system that leverages the presence of the nearby crowdsensors that monitor the same physical phenomenon, which we have shown improves the accuracy of the performance sensing application. Such an approach to calibration is particularly well suited to mobile crowdsensing scenarios in which the crowd senses and meets in public places.

Opportunistic crowdsensing shows a great potential to monitor the physical environment in a cost-effective and dynamic way by empowering people to contribute as part of their daily life. As such, crowdsensing is gradually becoming a cost-effective complement to fixed wireless sensor networks. The major emerging challenge currently facing mobile crowdsensing is related to the gathering and processing of an ever-increasing amount of data collected across time and space, with the intention of making the best sense of it so as to provide citizens and communities with relevant value-added information. In practice, people's mobility makes the crowdsensing contribution unevenly distributed over space and time, thus requiring the post-processing of sensing data on the cloud. As the number of contributors grows, the increasing number of observations that the crowdsensing platform must process becomes challenging: the high network and financial cost associated with a cloud-centric system hinders the widespread use of crowdsensing and the high computational cost required makes the modelling of the environmental phenomenon intractable. Given the growing volume of data involved to cover urban-scale areas, restraining the increasing operational cost of the cloud-assisted infrastructure and keeping to a minimum the resource consumed by the hand-held devices are a prerequisite. We have addressed this issue by exploiting the ever-increasing computing capacity of smartphones by evenly distribute the collection, interpolation and aggregation associated with the sensing data to powerful end devices. To this end we have introduced a crowdsensing middleware, which supports a novel in-network collaboration strategy that enhances the quality of the data before their transfer to the cloud, while reducing the related communication cost and resource consumption. The middleware tailors a set of utility functions that assess to what extent a device should carry out

a given crowdsensing task while achieving a trade-off between the benefit for all (*aka* for the group) and the related cost for the device. Rather than applying a naive combination (*e.g.*, averaging) of the data collected by the crowdsensors that would actually degrade the quality of the sensing data, our middleware performs an advanced inference. A Crowdsensor may establish a very precise interpolation of the regions it covers. Individual interpolations are then aggregated on the cloud, which renders the overall interpolation on the cloud much more tractable.

We observe a paradigm shift towards *"edgeness"* and information-centric networking that supports an early edge decision-making. In this perspective we have proposed a content-based publish/subscribe system that supports flexible and scalable messaging among *Things* (*e.g.*, sensors and actuators) and applications, delivering information across the wide range network to interested consumers as the amount of IoT devices increases and spreads geographically. We have built a cluster-based and hierarchical delivery structure that scales and tolerates failures while achieving both low latency and efficient use of resources. In the delivery structure, routers (a.k.a cluster heads) dedicate significant effort to filter and forward the messages (i.e., event notifications and (un)subscriptions). To reduce the routers loads we have proposed to help routers by including in the message the filters that need to be applied. In order to support such an approach without congesting the network, we have compacted filters. As a result, the flow of event notifications produced by any *Thing* can be filtered and distributed to provide situation-awareness and trigger actions accordingly.

In this manuscript I have presented a snapshot of my research as it stands today, keeping in mind that the development of a robust and resilient IoT remains a daunting task and many improvements are needed. From hardware to middleware, data collection to analysis and decision-making, actual practice to guiding theory, much works lies ahead in the creation of an efficient, resilient and ultimately life-enhancing IoT.

6.2 Perspectives

Conveying and processing the massive amount of data that is produced by the IoT – and in particular by crowdsensing applications – has become a major issue in terms of the management and operation of the underlying applications and networks. Conventional cloud computing systems and current communication architectures must be adapted and optimised to efficiently process the very large volumes of data generated by the IoT, which will continue to grow in time and space.

Further solutions and related optimisations need to be designed – acknowledging that some services can be moved close to smartphones and that much of the data can be stored or even analysed and filtered at an early stage to limit the amount of data transferred. Data, services and underlying infrastructure must also be secured in a complementary fashion. I intend to address these issues by exploring several research directions, ranging from the design of collaborative and distributed systems capable of supporting crowdsensing applications that operate on a large scale to the development of a secure environment for these applications.

Towards Virtual(ised) Crowdsensing

The data supplied by sensing applications, including mobile crowdsensing apps, contain insights about the physical and social environment and concerns user's activities and situation. Our past work has shown us that contributed data are often unusable unless first compared, fused, contextualised, enriched, classified *ect.*. Somehow, sensors are not intended to operate individually; they should not be considered as an autonomous pieces of hardwares and softwares capable of

monitoring a physical or social phenomenon on their own. Sensors are heterogeneous and operate at different scales spanning the nano, personal, and urban. Yet they are domain-specific and task-oriented: the mass of information they produce is specific to a sensor model, dedicated to a particular use(r), and mostly tailored to a given application with little or no possibility of use with other applications. To overcome this limitation we must abstract the set of physical sensors along with the operations performed on them to increase the data usability. The aim is to define and support virtual sensors that supply measurements that are not directly available by combining the measurements provided by a group of heterogeneous sensors. This also involves broadening the types of context we may thus characterise and integrating other sources of information of a social (social networks) or environmental (such as open street maps) nature. From this perspective it is crucial to define new methods of cooperation between physical and virtual sensors capable of performing complex and virtual tasks sustaining the enrichment, analysis, fusion, interpretation and transformation of data provided by both virtual and physical sensors, including other sources of relevant information. During the synthesis and comparison of the sensed information, the context/situation dependence and multi-dimensional nature of the sensed data cause a non-stationarity, which is not adequately addressed by conventional least square regression and derived interpolation methods. Rather than breaking down the data, using for instance multiple regressions to deal with the multi-dimensionality of the context, we propose an exploration of weighted regression [29] and moving weighted windows over the data, thereby estimating groups of coefficient values that fit the samples. This method first selects a bandwidth for an isotropic spatial weights kernel, typically a Gaussian kernel with a fixed bandwidth chosen by leave-one-out cross-validation. The choice of the bandwidth [28] is very demanding and requires investigation, as regressions must be fitted at each step.

Mobile and edge computing

Delivering seamless connectivity and communication to mobile users and ensuring sufficient quality of service despite the mobility of users/things remains a vast area of research, of which only certain challenges have been solved thus far. Our approach to improving communication reliability relies on exploiting multiple access networks that should be further adapted to deal with the dynamic aspects induced by mobility. As far as the mobile IoT is concerned, intelligent and opportunistic clustering helps to structure the network architecture; "intelligent gateway placement" techniques are prerequisites to enabling a smooth connection. Still there is a need to establish the most suitable path based on dynamic network properties (such as link quality) so as to convey packets to (mobile) gateways, taking into account the Braess paradox that often occurs as certain gateways act as vortexes. Supporting algorithms lead to an overuse of certain paths and therefore severe congestion, *e.g.*, with video [27]. Rather than merely selecting the optimal path our approach will aim at distributing/splitting the information using game theory. Although theory and algorithms for game have been extensively studied, a majority of the approaches [86] assumes that players are aware of the overall game state and previous moves before playing. This not realistic with IoT: *Things* (a.k.a players) have a limited observability and do share a consistent knowledge on their previous or ongoing actions. Incomplete knowledge introduces uncertainty that needs to be modelled [161], a critical issue that we address by finding an equilibrium without necessarily needing information regarding the presence of other data streams, the strategies implemented, or the paths taken by others. In addition, content fragmentation, network coding, joint and adaptive routing are all needed to effectively exploit the diversity of access paths while taking into account factors such as content specifics, user priorities and reliability.

On securing virtualised infrastructures

Networks are made up of virtualised infrastructures – ranging from cloud computing to NFV systems – that are constantly evolving and, as a result, highly complex and thus subject to vulnerabilities. Their dependability/security is contingent on our skill at detecting these vulnerabilities and in turn updating virtualised systems as soon as new threats are discovered. To this end it is necessary to understand how the attacker compromises the system, what vulnerabilities they exploit, what actions they undertake, and where vulnerabilities are located at the system level. Detecting and combating attacks in real time for virtualised infrastructures is tricky due to their modular structure, elasticity, rapid evolution (materialised by frequent upgrades), and containerization as isolated components comparable to black boxes.

While classification techniques are becoming popular to resolve problems related to anomaly detection [50], assuming the availability of a sufficiently large, diverse and labelled data set for a virtualised infrastructures is unrealistic. With this regards, unsupervised learning has to be privileged and new statistical profiling techniques using dynamic/multiple regressions need to be devised. Importantly, the model need to be pre-trained to properly reconstruct multivariate time series, using *e.g.*, stacked denoising auto-encoders [20]. This causes significant challenges due to the anomalies' variability and the need for designing adapted or dedicated models. Several sources of information should be collected and correlated in order to derive features, keeping in mind that the relationships between them are highly complex while taking into account real-time requirements. Rules must define the abnormal or normal character of a behaviour by comparing predicted and real behaviours. The simplest approach to setting a threshold based on (Euclidean) distance is likely to lead to many false positives, especially with noisy data. It is therefore crucial to design new methods based on statistical and probabilistic profiling approaches while also defining concrete policies to be applied to containers in order to create a line of defence that can be applied on a large scale and on the fly.

Bibliographie

Bibliography

[1] Mine project, http://cedric.cnam.fr/ sailhanf/murphy/.

[2] Mirai botnet, Dec. 2016.

[3] C. Adjih, D. Raffo, and P. Mühlethaler. Attacks against OLSR: distributed key management for security. In *OLSR interoperability workshop*, 2005.

[4] A. Adnane, C. Bidan, and R.T.S. Junior. Trust-based countermeasures for securing OLSR protocol. In *IEEE International Conference on Computational Science and Engineering (CSE)*, 2009.

[5] A. Adnane, R.T. Sousa, C. Bidan, and al. Autonomic trust reasoning enables misbehavior detection in OLSR. In *ACM ACM symposium on Applied computing*, 2008.

[6] A. Alattar, F. Sailhan, and J. Bourgeois. Reliable trust estimation in ad hoc networks using confidence interval. In *proc. of Conference on Security of Internet of Things (SecurIT)*, 2012.

[7] M. Alattar, F. Sailhan, and J. Bourgeois. Modeling and detecting intrusions in ad hoc network routing protocols. In *3SL Workshop, RENPAR, SYMPA, CFSE conferences*, 2011.

[8] M. Alattar, F. Sailhan, and J. Bourgeois. Log-based intrusion detection for manet. In *8th IEEE Wireless Communications and Mobile Computing Conference (IWCMC)*, 2012.

[9] Wi-Fi Alliance. Wi-fi peer-to-peer services (p2ps) technical specification (for wi-fi direct services certification). *Wi-Fi Alliance Document*, 2014.

[10] Dimitrios Amaxilatis, Evangelos Lagoudianakis, Georgios Mylonas, et al. Managing smartphone crowdsensing campaigns through the organicity smart city platform. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016.

[11] M.O. Asadoorian and D. Kantarelis. Essentials of inferential statistics. *University Pressof Amer*, 2005.

[12] I. Astic, C. Aunis, J. Dupire, V. Gal, E. Gressier-Soudan, C. Pitrey, M. Roy, F. Sailhan, M. Stimatic, A. Topol, and E. Zaza. Entre jeux pervasifs et applications critiques. In *Informatique et intelligence ambiante - des capteurs aux applications*, chapter 12. Hermes - Lavoisier, 2012.

[13] M. Astley, J. Auerbach, S. Bhola, and al. Achieving scalability and throughput in a publish/subscribe system. In *Research Report RC23103, IBM*, 2004.

[14] J. Baliosian, H. Oliver, A. Devitt, F. Sailhan, B. Danev, E. Salamanca, and G. Parr. Self-configuration for radio access networks. In *In proc. of the IEEE Workshop on Policies for Distributed Systems and Networks (POLICY)*, 2006.

[15] J. Baliosian, F. Sailhan, A. Devitt, and A.M. Bosneag. The omega architecture: towards adaptable self-managed networks. In *In proc. of IEEE Workshop on Distributed Autonomous Network Management (DANMS)*, 2006.

[16] G. Banavar, T. Chandra, B. Mukherjee, and al. An efficient multicast protocol for content-based publish-subscribe systems. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 1999.

[17] S. Banerjee, B. Bhattacharje, and C. Kommareddy. Scalable application layer multicast. *ACM conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2002.

[18] Avinoam Baruch, Andrew May, and Dapeng Yu. The motivations, enablers and barriers for voluntary participation in an online crowdsourcing platform. *Computers in Human Behavior*, 64, 2016.

[19] Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Elsevier Omega*, 34(06), 2006.

[20] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. *Advances in Neural Information Processing Systems 26 (NIPS'13*, 2013.

[21] K. Benson, C. Fracchia, Guoxi Wang, Qiuxi Zhu, and al. SCALE: Safe community awarenessand alerting leveraging the internet of things. *IEEE Communications Magazine*, 53(12), 2015.

[22] W. Berriche, F. Sailhan, and S. Secci. Prise en main du machine learning avec splunk. *Magasine de la cybersécurité offensive et défensive*, 2020.

[23] S. Bhattiprolu, E.W. Biederman, S. Hallyn, and al. Virtual servers and checkpoint/restart in mainstream linux. *ACM SIGOPS Operating Systems Review*, 42(5), 2008.

[24] S. Bianchi, P. Felber, and M. Gradinariu. Content-based publish/subscribe using distributed r-trees. In *European Conference on Parallel Processing (Euro-Par)*, 2007.

[25] H.B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communication with ACM*, 13(7), 1970.

[26] I. Bouacheria, Z. Bidai, B., and F. Sailhan. Leveraging multi-instance rpl routing protocol to enhance the video traffic delivery in IoMT. *Springer Wireless Personal Communications*, 2020.

[27] I. Bouacheria, Z. Bidai, B. Kechar, and F. Sailhan. Leveraging multi-instance rpl routing protocol to enhance the video traffic delivery in iomt. *Wireless Personal Communications*, 2020.

[28] A.W. Bowman and A. Azzalini. Applied smoothing techniques for data analysis. *Oxford University Press*, 1997.

[29] C. Brunsdon, A.S. Fotheringham, and M.E. Charlton. Geographically weighted regression: A method for exploring spatial nonstationarity. *Geographical Analysis, Wiley Press*, 1996.

[30] S. Burke. Regression and calibration. In *Statistics and data analysis*, LC.GC European online supplement, 2011.

[31] Vladimir Bychkovskiy, Seapahn Megerian, Deborah Estrin, and Miodrag Potkonjak. A collaborative approach to in-place sensor calibration. In *Information Processing in Sensor Networks*, pages 301–316. Springer, 2003.

[32] A. Campailla, D.S. Rosenblum, E. Clarkes, and al. Efficient filtering in publish-subscribe systems using binary decision diagrams. *ACM International Conference on Software Engineering (ICSE)*, 2001.

[33] Yanshuai Cao and David J Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions. In *Modern Nonparametrics 3: Automating the Learning Pipeline workshop at NIPS*, 2014.

[34] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and evaluation of a wide area event notification service. *IEEE Transactions on Software Engineering*, 19(1-6), 2001.

[35] M. Castro, P. Drushel, A. Kermarrec, and al. Scribe: a large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 2002.

[36] Xu Chen, Lingjun Pu, Lin Gao, et al. Exploiting massive d2d collaboration for energy-efficient mobile edge computing. *Wireless Communications*, 24(4), 2017.

[37] Yohan Chon, Nicholas D Lane, Fan Li, et al. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *ACM International Conference on Ubiquitous Computing*, 2012.

[38] M.U. Chowdhury, R. Doss, B. Ray, and al. Iot insider attack - survey. In *International Conference on Smart Grid and Internet of Things*, June 2020.

[39] T. Clausen and P. Jacquet. Optimized link state routing protocol. IETF RFC 3626, www.ietf.org, October 2003.

[40] Object management group. corba services – event service specification. technical report. https://www.omg.org/spec/evnt/1.1/pdf, 2001.

[41] A. Crespo, O.Buyukkokten, and H. Garcia-Molina. Efficient query subscription processing in a multicast environment. In *IEEE International Conference on Data Engineering (ICDE)*, 2000.

[42] Richard B Darlington and Andrew F Hayes. *Regression analysis and linear models: Concepts, applications, and implementation.* Guilford Publications, 2016.

[43] Pedro OS Vaz de Melo, Aline Carneiro Viana, Marco Fiore, et al. Recast: Telling apart social and random relationships in dynamic networks. *Performance Evaluation*, 87, 2015.

[44] P. Deutsch and J-L. Gailly. Zlib compressed data format specification version 3.3. IETF RFC 1950, 1996.

[45] K. Dickman. On the frequency of numbers containing prime factors of a certain relative magnitude. *Arkiv f. mat. Ast. Fys*, 44, 1930.

[46] Ngoc Do, Ye Zhao, Cheng-Hsin Hsu, et al. Crowdsourced mobile data transfer with delay bound. *ACM Transactions on Internet Technology*, 16(4), 2016.

[47] Trinh Minh Do and Daniel Gatica-Perez. Human interaction discovery in smartphone proximity networks. *Personal and Ubiquitous Computing*, 17(3), 2013.

[48] S.G. Doudane, F. Sailhan, and S. Collins. A clustering mechanism optimized for managing and wireless mesh networks. patent application number PCT/EP2007/052277, 2007.

[49] Norman R Draper and Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 2014.

[50] A. Drewek-Ossowicka, M. Pietrolaj, and J. Ruminski. A survey of neural networks usage for intrusion detection systems. *Journal of Ambient Intelligence and Humanized Computing, Springer*, 2020.

[51] Y. Du, V. Issarny, and F. Sailhan. User-centric context inference for mobile crowdsensing. In *Proc. ACM/IEEE IoTDI*, 2019.

[52] Yifan Du, Valérie Issarny, and Françoise Sailhan. User-centric context inference for mobile crowdsensing. In *ACM/IEEE International Conference on Internet of Things Design and Implementation*, 2019.

[53] Yifan Du, Valérie Issarny, and Françoise Sailhan. When the power of the crowd meets the intelligence of the middleware: The mobile phone sensing case. *ACM SIGOPS Operating Systems Review*, 53(1), 2019.

[54] Yifan Du, Francoise Sailhan, and Valerie Issarny. Let opportunistic crowdsensors work together for resource-efficient, quality-aware observations. In *IEEE International Conference on Pervasive Computing and Communications*, 2020.

[55] C. Duma, M. Karresand, N. Shahmehri, and al. A trust-aware, p2p-basedoverlay for intrusion detection. In *InDEXA Workshops*, 2006.

[56] M. Dunaway, R.R. Murphy, N. Venkatasubramanian, L. Palen, and D. Lopresti. Research agenda in intelligent infrastructure to enhance disaster management, community resilience and public safety. *CoRR abs*, 2017.

[57] Mostafa Elhoushi, Jacques Georgy, Aboelmagd Noureldin, and Michael J. Korenberg. A Survey on Approaches of Motion Mode Recognition Using Sensors. *IEEE Transactions on Intelligent Transportation Systems*, 18(7):1662–1686, July 2017.

[58] Martin Ester, Hans-Peter Kriegel, Jörg Sander, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM Conference on Knowledge Discovery and Data Mining*, 1996.

[59] L. Fallon, D. Parker, and F. Sailhan. Self-forming network management topologies. patent application number PCT/SE2006/070125, 2006.

[60] L. Fan, P. Cao, J. Ameida, and A.Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transaction on Networking*, 8(3), 2000.

[61] Katayoun Farrahi and Daniel Gatica-Perez. Discovering routines from large-scale human locations using probabilistic topic models. *Transactions on Intelligent Systems and Technology*, 2(1), 2011.

[62] S. Frickel, Sahra Gibbon, Jeff Howard, and al. Undone science: Charting social movement and civil society challenges to research agenda setting. *Science, Technology and Human Values*, 35(4), 2010.

[63] C.J. Fung, O.Baysal, J. Zhang, and al. Trust management for host-based collaborative intrusion detection. In *International Workshop on Distributed Systems: Operations and Management*, 2008.

[64] V. Gal, F. Sailhan, E. Gressier-Soudan, A. Topol, and J. Dupire. Entre jeux pervasifs et applications critiques. In *Informatique et Intelligence ambiante : Des capteurs aux applications ; Traité IC2, série Informatique et Systèmes d'Information*. Hermès Science, 2012.

[65] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: Current state and future challenges. *Communications Magazine*, 49(11), 2011.

[66] K. Garri, Sailhan, S. Bouzzefrane, and M. Uy. Anomaly detection in RFID systems. *International Journal on Radio Frequency Identification Technology and Applications, special issue on RFID-enhanced Technology Intelligence and Management*, 2010.

[67] H. Geng and R.van Renesse. Reliable broadcast for geographically dispersed datacenters. In *14th ACM/IFIP/USENIX International Middleware Conference*, 2013.

[68] Michele Girolami, Stefano Chessa, Gaia Adami, et al. Sensing interpolation strategies for a mobile crowdsensing platform. In *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2017.

[69] Michele Girolami, Stefano Chessa, Mauro Dragone, et al. Using spatial interpolation in the design of a coverage metric for mobile crowdsensing systems. In *IEEE International Symposium on Computers and Communication*, 2016.

[70] J. Goldszmidth and Y. Yemini. Distributed management by delegation. *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[71] Google(INC). Cloud pub/sub. https://cloud.google.com/pubsub/docs/overview.

[72] A. Granville. Smooth numbers: computational number theory and beyond. *Algorithmic number theory*, 44, 2008.

[73] Robert E Guinness. Beyond where to how: A machine learning approach for sensing mobility contexts using smartphone sensors. *Sensors*, 15(5), 2015.

[74] Bin Guo, Zhiwen Yu, Xingshe Zhou, et al. From participatory sensing to mobile crowd sensing. In *IEEE International Conference on Pervasive Computing and Communication Workshops*, 2014.

[75] J. Haerri, F. Filali, and C. Bonnet. Performance comparison of AODV and OLSR in vanets urban environments under realistic mobility patterns. In *IFIP Med-Hoc-Net workshop*, 2006.

[76] David Hasenfratz, Olga Saukh, and Lothar Thiele. On-the-fly calibration of low-cost gas sensors. In *European Conference on Wireless Sensor Networks*, pages 228–244. Springer, 2012.

[77] Takamasa Higuchi, Hirozumi Yamaguchi, and Teruo Higashino. Context-supported local crowd mapping via collaborative sensing with mobile phones. *Pervasive and Mobile Computing*, 13, 2014.

[78] A. Hildebrand. Integers free of large prime factors and the Riemann hypothesis. *Mathematika*, 31(2), 1984.

[79] P. Hilton, J. Pedersen, and J. Stigter. On partitions, surjections, and stirling numbers. Bulletin of the Belgian mathematical society, 1994.

[80] Y. Huang, D. Parkerand F. Sailhan, S. Collins, L. Fallon, and S.G. Doudane. Extensible optimized link state routing protocol. patent application number PCT/EP2006/21958, 2006.

[81] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 97–106. ACM Press, 2001.

[82] V. Issarny, G. Bouloukakis, N. Georgantas, F. Sailhan, and G. Texier. When service oriented computing meets the iot: A use case in the context of urban mobile crowdsensing. In *7th European Conference on Service oriented and Cloud Computing (ESOCC)*, 2018.

[83] Valerie Issarny, Vivien Mallet, Kinh Nguyen, Pierre-Guillaume Raverdy, Fadwa Rebhi, and Raphael Ventura. Dos and Don'ts in Mobile Phone Sensing Middleware: Learning from a Large-Scale Experiment. In *Proceedings of the 17th International Middleware Conference on - Middleware '16*, pages 1–13, Trento, Italy, 2016. ACM Press.

[84] H-A Jacobsen, A. Cheung, G. Lia, and al. The padres publish/subscribe system. *In Principles and Applications of Distributed Event-Based Systems. Hershey: IGI Global*, 2010.

[85] Hojjat Jafarpour, Sharad Mehrotra, Nalini Venkatasubramanian, and Mirko Montanari. Mics: an efficient content space representation model for publish/subscribe systems. In *ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2009.

[86] A. Jonnalagadda and L. Kuppusamy. A survey on game theoretic models for community detection in social networks. *Social Network Analysis and Mining*, 6, 2016.

[87] A. Josanga, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2), 2007.

[88] Oskar Juhlin and Mattias Östergren. Time to meet face-to-face and device-to-device. In *ACM Conference on Human-Computer Interaction with Mobile Devices and Services*, 2006.

[89] F.Z. Kaghat, F. Sailhan, and P. Cubaud. Application de la réalité augmentée sonore pour les visites de musées par les malvoyants. In *Handicap, congres sur les aides techniques pour les personnes handicapées*, 2012.

[90] Salil S Kanhere. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *International Conference on Distributed Computing and Internet Technology*. Springer, 2013.

[91] Merkouris Karaliopoulos, Orestis Telelis, and Iordanis Koutsopoulos. User recruitment for mobile crowdsensing over opportunistic networks. In *IEEE International Conference on Computer Communications*, 2015.

[92] T. Kohonen. Self-organised formation of topologically correct feature maps. *Biological Cybernetics*, 43(1), 1982.

[93] Linghe Kong, Mingyuan Xia, Xiao-Yang Liu, et al. Data loss and reconstruction in sensor networks. In *IEEE International Conference on Computer Communications*, 2013.

[94] Nicholas D Lane, Shane B Eisenman, Mirco Musolesi, et al. Urban sensing systems: opportunistic or participatory? In *ACM International Workshop on Mobile Computing Systems and Applications*, 2008.

[95] H.W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126(3), 1993.

[96] Chenguang Liu, Jie Hua, and Christine Julien. Scents: Collaborative sensing in proximity iot networks. In *IEEE International Conference on Pervasive Computing and Communications Workshops*. IEEE, 2019.

[97] Tony Liu, Jennifer Nicholas, Max M Theilig, et al. Machine learning for phone-based relationship estimation: The need to consider population heterogeneity. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(4), 2019.

[98] Z.-P. Lo and B. Bavarian. On the rate of convergence in topology preserving neural networks. *Biomedical Cybernetic*, 65, 1991.

[99] M. Alattar M. and F. Sailhan J. Bourgeois. On lightweight intrusion detection: modelling and detecting intrusions dedicated to OLSR protocol. *International Journal of Distributed Sensor Networks*, June 2013.

[100] J. M. Reyneri M. E. Hellman. Fast computation of discrete logarithms in GF(q). In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 3–13, 1983.

[101] A. Nelson Kyle Benson M. Y. S. Uddin, Guoxi Wang, Qiuxi Zhu, Qing Han, Nailah Alhassoun, Prakash Chakravarthi, Julien Stamatakis, Daniel Hoffman, Luke Darcy, and Nalini Venkatasubramanian. The scale2 multi-network architecturefor iot-based resilient communities. In *IEEE International Conference on Smart Computing (SMARTCOMP)*, 2016.

[102] Huadong Ma, Dong Zhao, and Peiyan Yuan. Opportunities in mobile crowd sensing. *Communications Magazine*, 52(8), 2014.

[103] Y. Madani, M. Erritali, J. Bengorram, and Sailhan. Social collaborative filtering approach for recommending courses in an e-learning platform. In *International conference on Ambient Systems, Networks and Technologies (ANT)*, 2019.

[104] Y. Madani, M. Erritali, J. Bengourram, and F. Sailhan F. A hybrid multilingual fuzzy-based approach to the sentiment analysis problem using sentiwordnet. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2019.

[105] J. Majewski and O. Boyko. Application of regression methods to humidity sensors calibration. *Proceedings of electrotechnical institute*, 252, 2011.

[106] Mahesh K. Marina, Valentin Radu, and Konstantinos Balampekos. Impact of Indoor-Outdoor Context on Crowdsourcing based Mobile Coverage Analysis. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges - AllThingsCellular '15*, pages 45–50, London, United Kingdom, 2015. ACM Press.

[107] Jan-Frederic Markert, Matthias Budde, Gregor Schindler, Markus Klug, and Michael Beigl. Private rendezvous-based calibration of low-cost sensors for participatory environmental sensing. In *Proceedings of the Second International Conference on IoT in Urban Space*, pages 82–85. ACM, 2016.

[108] Diego Mendez, Miguel Labrador, and Kandethody Ramachandran. Data interpolation for participatory sensing systems. *Pervasive and Mobile Computing*, 9(1), 2013.

[109] Sun Microsystems. Inc. jini(tm) jini distributed events specification v1.0. technical report, 2000.

[110] Budiman Minasny and Alex B McBratney. The matérn function as a general model for soil variograms. *Geoderma*, 128(3-4), 2005.

[111] G. Muhl. Large-scale content-based publish/subscribe systems. Ph.D. thesis, Darmstadt University of Technology, 2002.

[112] G. Muhl, A. Ulbrich, K. Herrmann, and al. Disseminating information to mobile clients using publish/subscribe. *IEEE Internet Computing*, 8, 2004.

[113] J.I. Munro and P.K. Nicholson. Succinct posets. *Algorthmica*, 2(76), 2012.

[114] Sudarsanan Nesamony, Madhan Karky Vairamuthu, and Maria E. Orlowska. On optimal route of a calibrating mobile sink in a wireless sensor network. In *IEEE Fourth International Conference on Networked Sensing Systems*, 2007.

[115] H. Nguyen, M.Y.S. Uddin, and Nalini Venkatasubramanian. Multistage adaptive load balancing for big active data publish subscribe systems. In *ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2019.

[116] J.-L. Nicholas. On highly composite numbers. In *Ramanujan revisited: proceedings of the centenary conference, University of Illinois at Urbana-Champaign, Ed. G. E. Andrews, B. C. Berndt, and R. A. Rankin). Boston, MA: Academic Press*, pages 215–244, 1998.

[117] M.E. O'Neill. The genuine sieve of eratosthenes. *Journal of Functional Programming*, 2008.

[118] P.Albers, O.Camp, J.Percher, and al. Security in ad hoc networks: a general intrusion detection architecture enhancing trust based approaches. In *WIS Workshop*, 2002.

[119] J.A. Patel, E. Rivière, I. Gupta, and A-M Kermarrec. Rappel:exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*, 53(13), 2009.

[120] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[121] N. Peng and S. Kun. How to misuse AODV: a case study of insider attacks against mobile ad-hoc routing protocols. *Ad Hoc Network*, 3(6), 2005.

[122] C. Perera, D.S. Talagala, C.H. Liu, and J.C. Estrella. Energy-Efficient Location and Activity-Aware On-Demand Mobile Distributed Sensing Platform for Sensing as a Service in IoT Clouds. *IEEE Transactions on Computational Social Systems*, 2(4):171–181, December 2015.

[123] P. Peris-Lopez, J.C. Hernandez-CastroJuan, J.M. Estevez-Tapiador, and A. Ribagorda. Rfid systems: A survey on security threats and proposed solutions. In *IFIP International Conference on Personal Wireless Communications*, 2006.

[124] P. Pietzuch and J.Bacon. Hermes: a distributed event based middleware architecture. *ACM International Conference on Distributed and Event-based Systems (DEBS)*, 2002.

[125] P.R. Pietzuch. Hermes, a scalable event-based middleware. Ph.D. thesis, University of Cambridge, 2004.

[126] C. Pitrey and F. Sailhan. On developping dependable positionning. In *3SL Workshop, CFSE conferences*, 2011.

[127] C. Pitrey and F. Sailhan. On developping dependable positionning. In *3SL Workshop, RENPAR, SYMPA, CFSE conferences*, 2011.

[128] C. Pomerance. The role of smooth numbers in number-theoretic algorithms. In *International Congress of Mathematicians*, pages 411–422, 1994.

[129] S. Ramanujan. Highly composite numbers. In *The Ramanujan Journal annoted by J.L. Nicolas and G. Robin, Kluwer Academic publishers*, pages 119–153, 1997.

[130] R. Rana, C.T. Chou, N. Bulusu, and al. Ear-Phone: A context-aware noise mapping using smart phones. *Pervasive and Mobile Computing*, 17:1–22, February 2015.

[131] Rajib Rana, Chun Tung Chou, Nirupama Bulusu, Salil Kanhere, and Wen Hu. Ear-phone: A context-aware noise mapping using smart phones. *Pervasive and Mobile Computing*, 17:1–22, 2015.

[132] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. journalthe MIT Press, 2006.

[133] G.F. Riley and T.R. Henderson. The ns-3 network simulator. In *Modeling and Tools for Network Simulation*. Springer Berlin Heidelberg, 2010.

[134] J.P.J Rodrigues, J.P. Pereira, and A. Aguiar. Impact of crowdsourced data quality on travel pattern estimation. In *ACM Workshop on Mobile Crowdsensing Systems and Applications (CrowdSenSys)*, pages 38–43. ACM Press, 2017.

[135] M. Rosenblatt. A central limit theorem and a strong mixing condition. *Proceedings of theNational Academy of Sciences of the United States of America*, 42(1), 1956.

[136] Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

[137] F. Sailhan, I. Astic, F. Michel, C. Pitrey, M. Uy, E. Gressier-Soudan, P. Gerbaud, and H. Forgeot. Conception d'une solution de localisation et de surveillance, à base de rfids actifs, défis et perspectives. In *INFORSID GEDSIP Workshop*, 2009.

[138] F. Sailhan, I. Astic, C. Pitrey, M.Uy, E. Gressier-Soudan, P. Gerbaud, and H. Forgeot. Sauvegarde du patrimoine en cas de sinistre : conception d'une solution de localisation et de surveillance a base de rfids actifs, défis et perspectives. In *GEDSIP Workshop*, 2009.

[139] F. Sailhan and J. Baliosian. An event prefiltering method for publish subscribe event system. patent application number PCT/EPT2007/056607, 2007.

[140] F. Sailhan and J. Baliosian. Scalable event notification for ubiquitous networks, ericsson internal report number lmi-07:000222 uen, 2007.

[141] F. Sailhan and J. Bourgeois. Log-based distributed intrusion detection for hybrid networks. In *Proc. of ACM Cyber Security and Information Intelligence Research Workshop (CSIIR)*, 2008.

[142] F. Sailhan, S. Collins, L. Fallon, D. Parker, S.G. Doudane, and Y. Huang. A policy driven grouping service for network management. patent application number PCT/EP2006/068448, 2006.

[143] F. Sailhan, E. Cuadrado-Salamanca, and J. Bourgeois. Group-based event notification in hybrid networks, research report rr2008-02, university of franche comté, 2008.

[144] F. Sailhan, T. Delot, A. Pathak, A. Puech, and M. Roy. Dependable sensor networks. In *INFORSID GEDSIP Workshop*, 2010.

[145] F. Sailhan, L. Fallon, K. Quinn, P. Farrell, S. Collins, D. Parker, S. Ghamri-Doudane, and Y.G. Huang. Wireless mesh network monitoring: Design, implementation and experiments. In *In proc. of IEEE Workshop on Distributed Autonomous Network Management (DANMS)*, 2007.

[146] F. Sailhan, Bourgeois J., and Issarny V. Security supervision in hybrid networks. *Studies in Computation Intelligence (SCI), Kacprzyk, Janusz ed., Springer*, 2008.

[147] F. Sailhan and M.O. Stehr. Folding and unfolding bloom filters - towards off-line planning and on-line optimization. In *proc. of the IEEE International Conference on Internet of Things (ITHINGS)*, 2012.

[148] Francoise Sailhan, Valérie Issarny, and Otto Tavares-Nascimiento. Opportunistic multiparty calibration for robust participatory sensing. In *Mobile Ad Hoc and Sensor Systems (MASS), 2017 IEEE 14th International Conference on*, pages 435–443. IEEE, 2017.

[149] Francoise Sailhan, Valérie Issarny, and Otto Tavares-Nascimiento. Opportunistic multi-party calibration for robust participatory sensing. In *IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, 2017.

[150] Matteo Saloni, Christine Julien, Amy L Murphy, et al. Lasso: A device-to-device group monitoring service for smart cities. In *IEEE International Smart Cities Conference*. IEEE, 2017.

[151] M. Sardar and K. Majumder. A new trust based secure routing scheme in manet. In *International Conference on Frontiers of Intelligent Computing: Theory and Applications*, 2013.

[152] Olga Saukh, David Hasenfratz, and Lothar Thiele. Reducing multi-hop calibration errors in large-scale mobile sensor networks. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 274–285. ACM, 2015.

[153] Luca Scalzotto, Kyle Benson, Georgios Bouloukakis, Paolo Bellavista, Valérie Issarny, Sharad Mehrotra, and Nalini Venkatasubramanian. An implementation experience with sdn-enabled iot data exchange middleware. In *ACM/IFIP/USENIX Middleware conference*, 2018.

[154] M. Sedighpour, M. Yousefikhoshbakht, and al. An effective genetic algorithm for solving the multiple traveling salesman problem. *Journal of Optimization in Industrial Engineering*, 34(08), 2012.

[155] B. Segall, D. Arnold, J. Boot, and al. Content based routing with elvin. In *Australian UNIX Users Group (AUUG)*, 2000.

[156] M. Smithson. Confidence intervales. *Sage University Papers Series on Quantitative Applications in Social Sciences*, 2003.

[157] M.J. Smithson. Statistics with confidence: an introduction for psychologists. *Sage Publications Limited*, 2000.

[158] Oasis standard. Mqtt version 5.0, march 2019.

[159] A. Standford-Clark and H.L. Truong. Mqtt for sensor networks (mqtt-sn), november 2013.

[160] M. S. Stankovic, S. S. Stankovic, and al. Asynchronous distributed blind calibration of sensor networks under noisy measurements. *IEEE Transactions on Control of Network Systems*, 5(1), 2016.

[161] M.O. Stehr, Minyoung Kim, and Tim McCarthy. A distributed computing model for dataflow, controlflow, and workflow in fractionated cyber-physical systems. *LNCS Computing with New Resources*, 2014.

[162] S. Tarkoma and J. Kangasharju. Optimizing content-based routers: posets and forests. *Distributed Computing*, 19(1), 2006.

[163] J. Tournier, F. Lesueur, F. Le Mouël, and al. A survey of iot protocols and their security issues through the lens of a generic iot stack. *Internet ofThings, Elsevier*, July 2020.

[164] C.H. Tseng, S. Tao, P. Balasubramanyam, and al. A specification-based intrusion detection model for OLSR. In *International Workshop on Recent Advances in Intrusion Detection*, 2008.

[165] Raphaël Ventura, Vivien Mallet, and Valérie Issarny. Assimilation of mobile phone measurements for noise mapping of a neighborhood. *Journal of the Acoustical Society of America*, 144(3), 2018.

[166] Raphaël Ventura, Vivien Mallet, and Valérie Issarny. Assimilation of mobile phone measurements for noise mapping of a neighborhood. *The journal of the acoustical society of America*, 144(3), 2018.

[167] Raphaël Ventura, Vivien Mallet, Valérie Issarny, et al. Evaluation and calibration of mobile phones for noise monitoring application. *Journal of the Acoustical Society of America*, 142(5), 2017.

[168] G. Vigna, S. Gwalani, K. Srinivasan, and al. An intrusion detection tool for AODV-based ad hoc wireless networks. In *Annual Computer Security Applications Conference (ACSAC)*, 2004.

[169] Leye Wang, Daqing Zhang, Yasha Wang, et al. Sparse mobile crowdsensing: challenges and opportunities. *Communications Magazine*, 54(7), 2016.

[170] Leye Wang, Daqing Zhang, Haoyi Xiong, et al. ecosense: Minimize participants' total 3g data cost in mobile crowdsensing using opportunistic relays. *Transactions on Systems, Man, and Cybernetics: Systems*, 47(6), 2016.

[171] Y. Wang, A. Yanget, and al. A deep learning approach for blind drift calibration of sensor networks. *IEEE Sensors Journal*, 17(13), 2017.

[172] Y.M. Wang, L. Qiu, D. Achlioptas, and al. Subscription partionning and routing in content-based publish/subscribe networks. *International Symposium on DIStributed Computing (DISC)*, 2002.

[173] K. Whitehouse and D. Culler. Macro-calibration in sensor/actuator networks. *Mobile Networks and Applications*, 142(8):463–472, 2003.

[174] Ian H Witten, Eibe Frank, Mark A Hall, et al. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[175] H. Xia, Z. Jia, X. li, and al. Trust prediction and trust-based source routing in mobile ad hoc networks. *Ad Hoc Networks*, 11(7), 2013.

[176] Yu Xiao, Pieter Simoens, Padmanabhan Pillai, et al. Lowering the barriers to large-scale mobile crowdsensing. In *ACM International Workshop on Mobile Computing Systems and Applications*, 2013.

[177] Sijia Yang, Jiang Bian, Licheng Wang, et al. Edgesense: Edge-mediated spatial-temporal crowdsensing. *IEEE Access*, 7, 2018.

[178] Zeromq distributed messaging, http://zeromq.org, 2015.

[179] Q. Zhu, F. Sailhan, Y.S.M. Uddin, V. Issarny, and N. Venkatasubramanian. Multi-sensor calibration planning in iot-enabled smart spaces. In *39th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019.

[180] S.Q. Zhuang and al. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. *ACM Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 2001.