Protocols and Architectures of IoT Networked Systems for Smart Spaces and Crowd Environments

Kuendig, Stéphane Jean-Jacques

This publication URL: https://archive-ouverte.unige.ch/unige:156388
Publication DOI: 10.13097/archive-ouverte/unige:156388

UNIVERSITÉ DE GENÈVE FACULTÉ DES SCIENCES

Département d'informatique Professeur José Rolim

Protocols and Architectures of IoT Networked Systems for Smart Spaces and Crowd Environments

THÈSE

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention informatique

par

Stéphane Jean-Jacques KUENDIG

de Genève, Suisse

Thèse N^{o}5605

GENÈVE
2021

Anyone who stops learning is old, whether at twenty or eighty. Anyone who keeps learning stays young. The greatest thing in life is to keep your mind young.

— Henry Ford

Acknowledgements

The research leading to this thesis was conducted at the TCS-Sensor Lab at the University of Geneva. First of all, I would like to express my deepest gratitude to my supervisor prof. Jose Rolim for giving me the opportunity to work in this lab under his guidance. He has given me the freedom to explore subjects of my interest, trusting me with important work and advising me all along the way. I would also like to thank Dr. Pierre Leone with whom collaboration has not only been very fruitful for my research, but also highly enjoyable thanks to his kind and supportive character. Special thanks to Dr. Marios Angelopoulos for his precious advice and support at crucial points throughout this journey, without which this thesis would be significantly impoverished.

Furthermore, I would like to thank former TCS-Sensor Lab members Dr. Kasun Samarasinghe and Dr. Orestis Evangelatos for helping me integrate during the first steps of my PhD, as well as my fellow PhD students at the Computer Science department of the University of Geneva, Simon Senécal, Sahar Aljalbout and Yvain Tisserand for making the workspace a friendly as well as productive environment.

Finally, my most heartfelt thanks to all the friends and family who have been by my side throughout the years, in the never-ending path of knowledge and education.

Stéphane Kündig

Abstract

Networked systems play a major role in the modern world, enabling connectivity between computers, mobile phones and lately even sensor devices of very small size. These low-power, often ad-hoc, networks exhibit particular characteristics and are bound to specific challenges, which have been the object of study of fields such as ubiquitous computing, Wireless Sensor Networks (WSN) and the Internet of Things (IoT). This thesis is focused on such low-power networked systems in the context of smart spaces and crowd environments. The goal is to define efficient protocols and architectures given the subtleties of these settings, addressing issues such as decentralization of tasks, handling of heterogeneous resources and dealing with crowd unreliability. The challenges addressed are met throughout the whole lifecycle of a system's creation, from architecture design and definition of requirements, to implementation and finally experimental evaluation. The thesis starts by examining the purely networking aspects, investigating the field of mobile ad-hoc networks and proposing a novel routing algorithm which disseminates path using a loop-erased random walk. Then, the focus is shifted to smart spaces and WSN, with the implementation of two fully-fledged IoT networked systems, the first providing location-based automation to mobile users and the second modelling and augmenting a WSN testbed's resources. Both systems are deployed and tested at the Syndesi WSN testbed of the TCS-Sensor lab, at the University of Geneva. In the third part of the thesis the focus is directed on crowdsourced systems, starting with a study defining the formal characteristics, architecture and requirements of IoT-related crowdsourced systems. Subsequently, elements from the novel field of edge computing are put into play, in an attempt to implement edge computing as a crowdsourced system. Finally, the findings and insights gained throughout the previous parts are put together in a crowdsourced framework enabling collaborative computing in local neighborhoods, by forming smartphone ad-hoc networks and providing a task sharing mechanism. The framework is evaluated in a prototype application developed for Android OS, compatible with more than 99% of Android devices presently owned by users, achieving very good results thus marking a successful proof-of-concept.

Keywords: Decentralized Networked Systems, System Architecture, Ad-hoc Networks, Routing Protocols, Internet of Things, Crowdsourced Systems

Résumé

Les systèmes en réseau jouent un rôle majeur dans le monde moderne, permettant la connectivité entre les ordinateurs, les téléphones mobiles et récemment même les capteurs de très petite taille. Ces nouveaux réseaux à basse consommation, souvent formés de façon ad hoc, présentent des caractéristiques particulières et sont liés à des défis spécifiques, qui ont fait l'objet d'étude dans des domaines tels que le ubiquitous computing, les réseaux de capteurs sans fil (Wireless Sensor Networks - WSN) et l'internet des objets (Internet of Things - IoT). Cette thèse se concentre sur de tels systèmes en réseau à basse consommation, dans le contexte des smart spaces et des environnements de foule. L'objectif est de définir des protocoles et des architectures efficaces compte tenu des subtilités de ces environnements, en abordant des questions telles que la décentralisation des tâches, la manipulation de ressources hétérogènes et la gestion du manque de fiabilité de la foule. Les défis abordés sont relevés tout au long du cycle de vie de la création d'un système, depuis la conception de l'architecture et la définition des exigences, jusqu'à la mise en œuvre et enfin l'évaluation expérimentale. La thèse commence par examiner les aspects purement réseau, en étudiant le domaine des réseaux mobiles ad-hoc et en proposant un nouvel algorithme de routage qui dissémine des chemins en utilisant un random walk qui élimine les boucles. Ensuite, l'accent est mis sur les smart spaces et les WSN, avec la mise en œuvre de deux systèmes IoT complètes, le premier réalisant l'automatisation des appareils de bureau en fonction de la localisation des occupants et le deuxième modélisant et augmentant les ressources d'un WSN testbed. Les deux systèmes sont déployés et testés sur le WSN testbed Syndesi du laboratoire TCS-Sensor, à l'Université de Genève. Dans la troisième partie de la thèse, l'accent est mis sur les systèmes crowdsourcés, en commençant par une étude définissant les caractéristiques formelles, l'architecture et les exigences des systèmes crowdsourcés liés à l'IoT. Ensuite, des éléments du nouveau domaine de l'edge computing sont mis en jeu, dans une tentative d'implémenter l'edge computing comme un système crowdsourcé. Finalement, les conclusions et les idées acquises tout au long des parties précédentes sont rassemblées dans une implementation d'un système de collaborative computing pour les occupants d'une région locale, en formant des réseaux ad-hoc de smartphones et en fournissant un mécanisme de partage des tâches. Le système est évalué dans un prototype d'application développé pour Android, compatible avec plus de 99 % des appareils Android actuellement détenus par les utilisateurs, avec de très bons résultats, ce qui constitue une preuve de concept réussie.

Keywords: Systèmes Décentralisés, Architecture des Systèmes, Réseaux Ad Hoc, Protocoles de Routage, Internet des Objets, Systèmes Crowdsourcés.

Contents

List of Figures

List of Tables

Part I

Introduction

1 Introduction

1.1 Overview

In the history of human civilisation, human societies have always strived for efficient ways of achieving connectivity between their members. From hydraulic semaphores used in ancient Greece, to smoke signals and drums in Africa and the Americas, telecommunications have played a major role in the evolution of societies. The use of electric signals in telecommunications, starting in the 18th century with the telegraph, brought a never-seen before revolution to the field, which gradually led to the creation of the telephone, radio, and eventually computer networks. Wired or wireless, computer networks are abundant in the modern world, enabling extreme levels of connectivity. Indicatively, the largest network in the world, the internet, to this day interconnects almost 60% of the world's population and more than 90% in the developed world [9].

More than simply providing a mean of communication, in the last decades networks have evolved into systems aiming to improve quality of life in various ways. With the inclusion of nodes equipped with sensing capabilities, a network can monitor its environment and act upon eventual changes, while also gather data that can serve some purpose after post-processing. Provided a gateway to the internet, the collaboration and integration between multiple such systems becomes possible, independent of their physical distance, allowing for larger scale coordination. This paradigm, referred to as the Internet of Things (IoT), has been evolving fast during the past two decades as novel technologies in embedded systems and low power networks are sprouting, enabling more and more objects to be included in the IoT ecosystem.

With regard to such next-generation networks, a key domain that has also seen rapid development in recent years is the one of personal electronic devices. Over the years, traditional cell phones have been enhanced in hardware and software, gradually evolving into smartphones, with cellular networks following along and starting to accommodate internet-providing features. In addition, affordable Single Board Computers (SBCs) for personal use, highly equipped with sensing and networking tools, have begun to enter the market. All of the above has led to an augmented IoT paradigm, which includes mobile users, and has a more decentralised nature. Furthermore, the appearance of a large, diverse set of connected users in this augmented paradigm, has promoted the practice of crowdsourcing, be it on tasks that require human intelligence or in mere gathering of crowd devices' computing resources.

The goal of this thesis is to define new methods, methodologies, efficient protocols and architectures, for such state of the art networked systems, in the context of smart spaces and crowd environments. The problems addressed are met throughout the whole lifecycle of a system's creation, from architecture design and definition of requirements, to implementation and finally experimental evaluation. The thesis starts by focusing at networking aspects such as routing protocols, before moving towards investigating fully-fledged networked systems that are implemented in a Wireless Sensor Network (WSN) testbed, located at the University of Geneva. Finally, the focus is directed to crowdsourced systems, with contributions in terms of the formal definition of their characteristics, architecture and requirements in the context of IoT, as well as the design and development of prototype applications.

1.2 Contributing Publications

The content of this thesis is based on the work appearing also in five articles published on scientific venues, as well as a work contributing to an International Telecommunications Union (ITU) published standard. The chapters of the thesis are not carbon copies of the published works, as their content has been updated, adjusted and expanded with respect to present findings. Following are the details of the above-mentioned publications:

1. *A Distributed Algorithm Using Path Dissemination for Publish-Subscribe Communication Patterns*, Stéphane Kündig, Pierre Leone, José D. P. Rolim, Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access, MobiWac 2016, Malta, November 2016.

2. *Indoor Location for Smart Environments with Wireless Sensor and Actuator Networks*, Zhongliang Zhao, Stéphane Kündig, Jose Luis Carrera, Blaise Carron, Torsten Braun, José D. P. Rolim, 42nd IEEE Conference on Local Computer Networks, LCN 2017, Singapore, October 2017.

3. *Modelled Testbeds: Visualizing and Augmenting Physical Testbeds with Virtual Resources*, Stéphane Kündig, Constantinos Marios Angelopoulos, José D. P. Rolim, Proceedings of the International Conference on Information Technology & Systems (ICITS 2018), Ecuador, January 2018

4. *Y.4205: Requirements and Reference Model of IoT-related Crowdsourced Systems*, Study Group 20: “Internet of things (IoT) and smart cities and communities", Question 5/20: "Research and emerging technologies, terminology and definitions”, February 2019

5. *Crowdcloud: a Crowdsourced System for Cloud Infrastructure*, Mahmood Hosseini, Constantinos Marios Angelopoulos, Wei Koong Chai, Stéphane Kündig, Cluster Computing 22, Springer, January 2019

6. *Crowdsourced Edge: A Novel Networking Paradigm for the Collaborative Community*, Stéphane Kündig, Constantinos Marios Angelopoulos, Sanmukh R. Kuppannagari, José D. P. Rolim, Viktor K. Prasanna, 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), June 2020.

1.3 Thesis Structure

The thesis is divided in three parts: Part I delves into the networking aspects of wireless ad hoc networks, examining the challenges in finding efficient routing protocols in that context. In this light, a novel algorithm is proposed attempting to tackle some of these challenges, by using path dissemination via a custom loop-erased random walk. Part II addresses the field of WSN-enabled smart spaces, presenting two fully-fledged networked systems enabling i) location-based automation and ii) resource visualisation and augmentation. Both systems are implemented at a prototype level and experimentally evaluated in a WSN testbed located in the TCS-Sensor lab, at the University of Geneva. In Part III, a series of works in the domain of crowdsourced systems is presented. First, a formal definition of IoT-related crowdsourced systems is provided, together with their requirements and canonical architecture. Subsequently, the convergence of crowdsourced systems with the novel field of edge computing is investigated, leading to the definition of a crowdsourced edge computing paradigm, followed by an extensive concretisation use-case. In the final chapter, a framework connecting user smartphones and enabling collaborative computing is implemented, enabled via a novel data collection protocol that we define, which is evaluated in a prototype mobile application developed for Android OS. Finally, Part IV concludes the thesis and discusses the next steps.

Part II

Network Routing in Wireless Ad Hoc Networks

2 A Distributed Algorithm for Rendez-vouz Routing Using Path Dissemination

2.1 Introduction

Wireless networks can be classified into two distinct types: infrastructured and infrastructureless [10]. Infrastructured wireless networks ensure connectivity via some designated central nodes called Access Points (APs), which perform the message routing and are typically more enhanced in hardware capabilities (hence some infrastructure already in place is required). On the contrary, infrastructureless wireless networks, also referred to as wireless ad hoc networks, do not rely on any predefined infrastructure, rather they utilise each node's individual resources for performing all necessary network functions. It is evident that ensuring connectivity and achieving robust message routing is particularly harder in infrastructureless networks. The design of efficient routing protocols in such settings is particularly difficult, with factors such as constraints in node resources and energy supplies, as well as frequent changes in network topology, imposing significant challenges [11].

Looking at routing protocols in wired networks, two general approaches are dominant: link-state [12] and distance-vector [13] routing. In link-state routing, every node periodically calculates the link-state costs of its neighbouring nodes and broadcasts it to the network using a flooding mechanism. Using these messages, every node constructs a graph of the network connectivity which is used to calculate optimal routing paths. In distance-vector routing, nodes build routing tables with distance information, from which optimal paths are chosen based on hop distance. Again, a periodic exchange of messages between nodes is necessary to accurately map and maintain the network routes. The above protocol families, although widely used and highly developed, do not perform well in wireless networks, especially of large scale [14]. This is is because regular route updates in large networks can consume significant part of the bandwidth which is limited, while also the energy toll on wireless nodes can impose too frequent power recharging. In fact, in some cases finding and maintaining routes to every node in the network can be thought as a resource overkill, as each node will communicate only with a subset of the network. A more efficient approach consists in building a hierarchical routing scheme, where the scalability issue is tackled using the divide-and-conquer principle. Another category of protocols designed to cope with the above problems is reactive or on-demand protocols, in which routes are calculated only when they are required by a source, as opposed to the proactive protocols where all routes are calculated on startup. Hybrid protocols also exist in that context, combining basic properties from both categories.

In this work, we consider a large set of wireless nodes with only knowledge of the local neighborhood and no routing or overlay layer yet in place. We focus on finding an efficient way to build the overlay layer together with the routing layer, being inspired from publish/subscribe systems. In pub/sub systems, a publisher node disseminates a message advertising some information/service (publication) while subscriber nodes disseminate corresponding requests (subscription). When there is a publication/subscription match in the network a permanent link is established. In our method we use this communication pattern to build routing paths and interconnect the whole network. The key difference is that we don't make use of broker nodes, i.e. specialized central nodes that have increased network knowledge and help message routing, which is almost always the case in pub/sub systems. In the next section, we underline the design criteria underpinning our solution and we discuss related work that tackle the same problem, before presenting our algorithm in detail together with experimental results.

2.2 Design principles

We assume a network of wireless nodes of very large scale, where each node has limited computing and storage capacity as well as energy supplies. The key ingredient of an efficient solution lies in finding a robust dissemination algorithm to disseminate information in the network. We emphasize that:

- The simplicity of the algorithm is important in order to limit energy consumption due to computation and message exchange.
- At every stage only local network knowledge is assumed, hence routing decisions are based solely on neighboring node states.

Our approach consists in following a publish/subscribe communication pattern to build routing paths. Each of the publisher and subscriber nodes disseminate the information using a distributed algorithm which is based on the Laplacian random walk [15] and is detailed later. Once there is a pub/sub match, following a path intersection, a route is established. The criteria that should be met and that will be used to evaluate the efficiency of the algorithm are the following:

- Two paths should intersect with a high probability.
- The time to intersection is important as it can give an estimate of when the path dissemination should stop. It should not depend too much on the originating nodes.
- The dissemination paths must balance the node congestion. That is because the number of paths that can pass through a single node is limited due to memory constraints.

Finally, to complete the list of the principles that regulate the design our system we should mention that large scale wireless systems are likely to be composed of very heterogeneous nodes that work asynchronously. Moreover, the communication paradigm should be robust and tolerate disruption or variable delays. In the next section we go over similar approaches that exist in the literature that try to tackle the above problems.

2.3 Related Work

There exists robust dissemination algorithms such as rumor routing [16], directed diffusion [17], gradient-based routing [18] or Pastry [19] that are often used in similar context. Based on these algorithms, higher-level middleware frameworks have been built such as Scribe [20], Mires [21] or MQTT [22] which have been popularized in applications in the Internet of Things (IoT). Nevertheless, all of the above approaches to some extent make use of broker nodes, which is something we want to avoid in our case. In fact, we want to avoid the construction and management of a subnetwork of nodes, for instance as a tree as in [23], based on the principle that in order to maximize the lifetime of a network of limited energy resources, load balancing is paramount [24, 25]. Moreover, such systems are built on the top of the routing algorithm layer and in our case, we do not assume this layer's existence.

Data-oriented network architectures such as information-centric networking (ICN) [26] or content-centric networking [27] follow a similar approach but most of the times energy-related matters are ignored, as fixed, resourceful nodes are often assumed. Adaptations of ICN for wireless sensor networks (WSN) attempt to tackle the energy issue, as for instance the work in data-centric routing [28] which tries to deal with it using optimal data aggregation.

The semi-probabilistic approach in [29] is in the same spirit with our work. In that work, a wireless broadcast is used as the communication primitive. The approach is called semi-probabilistic because subscriptions are flooded in the network and the flood is controlled by the subscription horizon that limits the number of message repetitions. The dissemination of publication is probabilistic, in the sense that it is a broadcast but only a fraction of the nodes relay the message. The properties that are common to our approach are that: i) all the nodes are potentially broker nodes and ii) the process may fail.

We mention as well the work in [30] where the authors focus on the problem of building paths that intersect. The scheme is called Double Rulings and is based on the same heuristic that two lines on the plane intersect, i.e. propagates publications and subscription along two lines in the plane and match the two at the node where the lines intersect. Their approach is much more geometrically-dependent than ours. In particular, the method presented in this study can be applied on any graph topology and requires only the (local) abstract description of the graph.

2.4 Algorithm Description

For the the demonstration of the algorithm we employ a publish/subscribe system. The high-level idea is that publisher and subscriber nodes emit messages which traverse the network, while piggy-backing their route, and once there is a match the merged path from both sides is confirmed as an established route.

In order to detail the algorithm we define the following notation:

- The message m with the fields:
  - *Type*, which takes the values *pub*, *sub* or *confirm*
  - *Info*, where the nature of the advertised/requested information is stored
  - *Piggypath*, where the traversed route of the message is stored in sequence
- The routing table RT with the fields:
  - *Info*, where the m.info is stored
  - *From*, with the address of the message originator
  - *Confirmed*, which is yes/no depending if the path is confirmed

Info	*From*	*Confirmed*	*Type*
Pizza	10.0.1.1	NO	pub
Parking	10.0.1.2	NO	sub
Hotel	10.0.1.3	YES	-
.	.	.	.
.	.	.	.
.	.	.	.

Table 2.1: An example routing table of a node that has received one publication and one subscription message, and contains a confirmed path (route).

- *Type*, where the type of the entry is stored (pub/sub or left empty if path is confirmed)

and the methods:

- *RT.add*, which adds a new line in the table
- *RT.check*, that accepts an argument of type *Info* and an argument of type *Type*, and returns true/false if a line with a matching information and type exists or not in the RT.
- *RT.available()*, which returns true/false depending on if there is available space in the RT.

In the following figures, we depict the algorithm pseudocode as well as a flowchart detailing the node's behaviour upon the reception of messages of type *sub* and *pub*.

```
1: Upon reception of m of type sub
2: if (RT.check(m.info,sub)) then                    ▷ check if info on RT
3:     Compute next node and forward message
4: else if (RT.check(m.info,pub)) then               ▷ check if pub/sub match
5:     Matching occurs. Confirm both paths.
6: else if (RT.available()) then                     ▷ if RT is full the process aborts
7:     RT.add(m.info,m.sender,sub)                   ▷ the state is not confirmed in RT
8:     Compute next node and forward.
9: end if
```

Figure 2.1: Algorithm pseudocode upon reception of a message m of type sub.

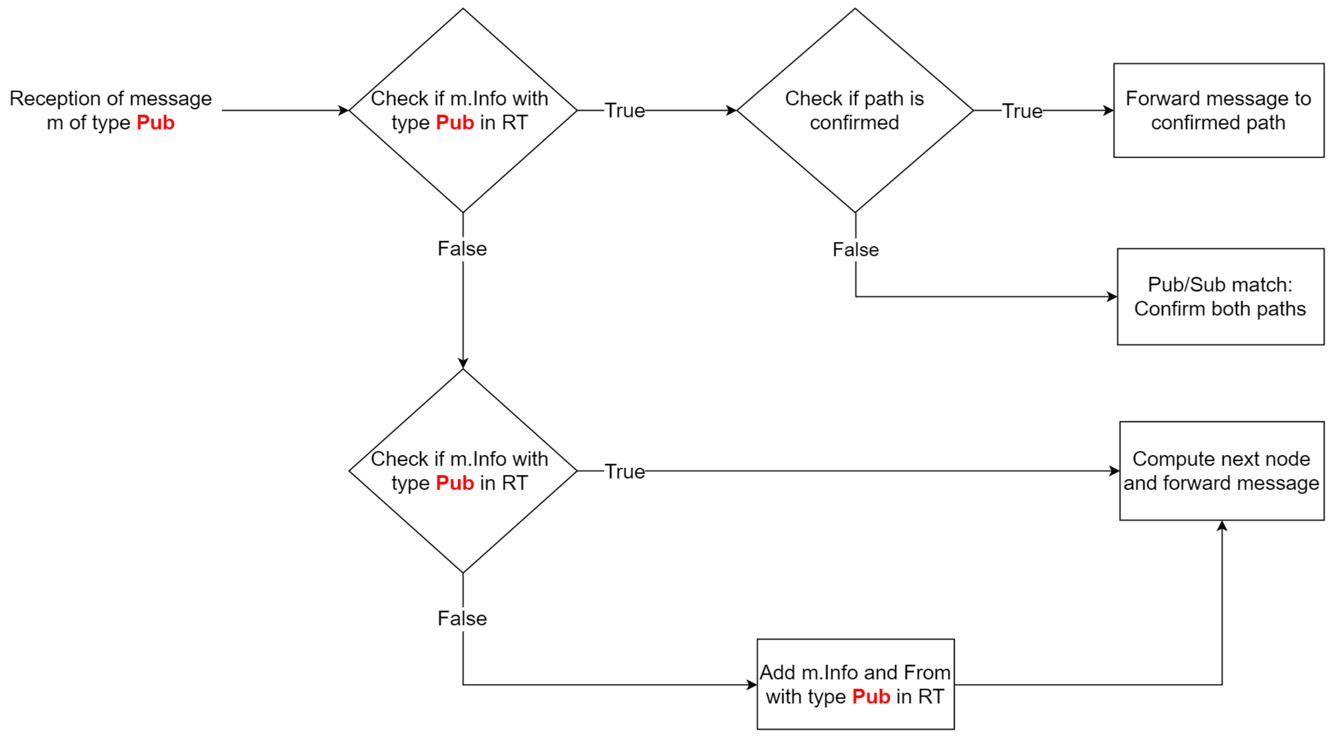

Reception of message m of type Pub
Check if m.Info with type Pub in RT
True
Check if path is confirmed
True
Forward message to confirmed path
False
False
Pub/Sub match: Confirm both paths
Check if m.Info with type Pub in RT
True
Compute next node and forward message
False
Add m.Info and From with type Pub in RT

Figure 2.2: Flowchart showing the sequence of actions upon a reception of a message of type *pub*.

In fact, as we can see, the behavior is symmetrical at the reception of a publication or subscription message. If there is a match the two paths are confirmed into an active route, and if not the message is forwarded in the network, after its content has been stored locally in a routing table. The choice of the next node to forward the message is where the algorithm difficulty lies; the created paths should have high probability to intersect and the load must be sufficiently balanced for the solution to be viable. In the following section, we detail the heuristic used which is based on a Laplacian random walk [15].

2.5 Computation of the Next Node in the Path

When a node u forwards a message in the network the following process is applied to decide the next recipient. We assume that node u possess local knowledge of its d-th neighborhood, i.e. all the nodes at hop distance less or equal than d. We denote $N_d(u)$ to be the dth neighborhood of u. Similarly, u knows the d last nodes that belong to the path traversed so far, a set that we denote as $P(u)$. As a message traverses the network, $P(u)$ is piggybacked on its memory, with the set being updated each time at a new node arrival, while the older node in the set is removed. For each node in its neighborhood $v \in N_d(u)$, node u computes the (harmonic) function $h(v)$ given by:

$$h(v) = \begin{cases} 1 & \text{if } v \in P(u) \bigcup \{u\}, \\ \frac{1}{deg(v)} \sum_{w \in P(u), w \sim v} h(w) & \text{else.} \end{cases} \tag{2.1}$$

where $v \sim w$ refers to the neighboring nodes of v and $deg(v)$ to the number of neighbors of v. The harmonic function $h(v)$ calculates the average of values $h(w)$, $w \sim v$, with the border conditions $h = 1$ at nodes belonging to the path (last d steps or u) and $h = 0$ at nodes that do not belong to $N_d(u)$. The value $h(v)$ expresses the probability that a random walk that started at v will hit the path before going outside of $N_d(u)$. Therefore, the next node to forward is selected such as:

$$\text{next node } (u) = \text{argmin}\Big\{h(v) \mid v \sim u\Big\}. \tag{2.2}$$

The intuition behind the selection of the next node stems from the following reasoning. There are two problems to solve: first, we want to disseminate the information as efficiently as possible (**dissemination problem**), and second, we want to reach a node which has been visited by another path (pub/sub match), (**discovery problem**).

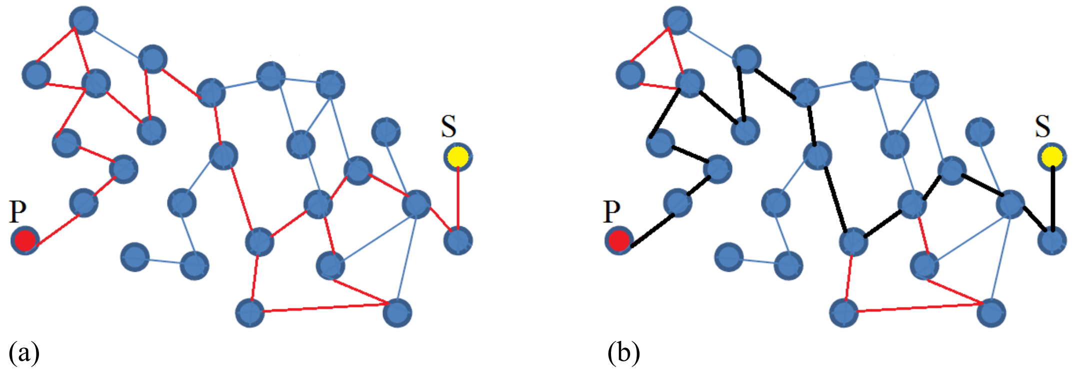


Figure 2.3: Example of path dissemination from P to S using a simple random walk. (a) Initial path and (b) path after erasing the loops (black). The loop-erased path can be produced choosing at every stepm the min of the harmonic function h with the border conditions $h = 1$ at P and $h = 0$ at S.

The solution to the first problem consists in building paths without loops, thus without revisiting a specific node more than once. Taking a produced path from a simple random walk (which is also good for balancing the load), we can get its loop-erased version using the Laplacian random walk with border conditions 1 at the start and 0 at the end (fig. **??**). In our case, as we don't possess full knowledge of the network we adapt the Laplacian random walk to a local version, with the border conditions at the radius of the known N_d neighborhood. It turns out that this heuristic optimizes the discovery problem as well; in fact, to increase the probabilities of discovery a node u must choose a next node v which is likely to have been traversed by another path. Node u knows that another path cannot have gone through u or $P(u)$, since in that case intersection would have occurred. This can be expressed with a probability that a random walk starting from v has hit u or $P(u)$, using a harmonic function $h(v)$ like the one in Equation (2.1). If this probability is large then it is unlikely that v was traversed by another path. Therefore, it is more efficient to pick the node with the smaller probability $h(v)$ as the next step of the path, which is what is proposed from our heuristic.

2.6 Experimental Results

To evaluate the efficiency of the algorithm a set of simulations was performed. The parameters of the simulations are denoted as follows:

- Network size N, taking values in the range of $[1500, 5000]$
- Network connectivity radius r, which creates the network links based on unit disc graph connectivity. r was kept at a constant value of $r = 0.04$ which means that larger size networks were proportionally denser, i.e. from 7.5 up to 25 neighbors per node in average.
- Depth d of local network knowledge, expressed in number of hops. d was given small values $\{1, 2, 3\}$.
- Algorithm upper steps boundary USt_b which was set to 500.
- Node congestion boundary C_b, limiting the maximum number of paths that can pass through a single node. C_b reflects to the size of the node's routing table RT that, when full, the node refuses further connection.

In each simulation a N-size connected network was created with its nodes distributed randomly and uniformly in a fixed 1×1 area, following a unit disk graph connectivity of disk radius r. The path building algorithm was then applied to a random permutation of node pairs, i.e. each node played the subscriber and publisher role exactly once, which led to the building of a maximum of N paths with source-destination pair drawn randomly with uniform probability. The above process is commonly used to evaluate routing algorithms [31].

The building of the publisher and subscriber paths happened in a synchronous manner, such as for each algorithm iteration one step was added alternately to each path. Asynchronicity, as discussed in [32], is an important parameter but we chose not to evaluate it at this stage of our work in order to better focus on the other parameters. The upper steps boundary of $USt_b = 500$, after which attempt for intersection was abandoned, limits respectively the created path length, denoted as a l, to a total of $l = 1000$.

Based on the simulation results, the algorithm was evaluated on the following criteria:

1. **Probability of intersection**. If two paths start at nodes u, v (randomly and uniformly selected) and evolve synchronously what is the probability that they intersect before a step boundary $St_b(\leq USt_b)$.
2. **Time to intersection**. We assume that one step is added to a path in one unit of time, so time to intersection is the total of algorithm iterations performed until intersection occurs.
3. **Network congestion**. The percentage of nodes of the network that are congested with respect to C_b, after the end of an algorithm run.
4. **Path length**. Path length l refers to the length of the path that links the publisher and the subscriber upon intersection. It is the sum of the publisher's and the subscriber's path until the intersection point and after erasing the possible loops.

2.6.1 Probability of Intersection

We estimate the probability of intersection based on the percentage of successful publisher-subscriber path intersections in a complete network permutation, with u, v selected randomly and uniformly. In the graphs that follow, we plot the results of this success rate percentage with St_b in the range of $[25, 500]$, while alternating the other algorithm parameters.

Initially, we explore the impact of depth d on the probability of intersection. Figure 2.4 shows the effect of d in the discrete value range $1, 2, 3$ for constant values of

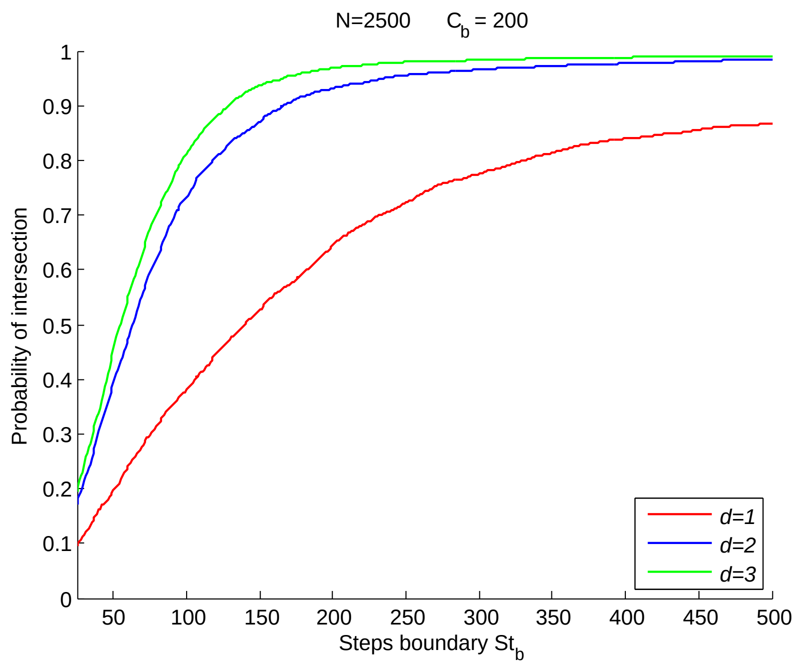


Figure 2.4: Probability of intersection as a function of steps boundary for different values of depth d.

$N = 2500$ and $C_b = 200$. We observe a significant improvement in the transition from $d = 1$ to $d = 2$, but only a marginal one from $d = 2$ to $d = 3$. Simulations with even larger values of d indicated that a further increase has a negligible marginal impact on the success rate, compared to the increase in complexity it comes with. Therefore, from now on we proceed by presenting simulation results with depth $d = 3$, having arrived to the conclusion that this setting provides the best value over effort ratio.

In terms of network size, Figure 2.5 demonstrates the algorithm's success rate as the network size grows larger, all other things held constant. Interestingly, adding more nodes facilitates the intersection of paths to some extent, as can be seen on the graph for N in $[1500 - 3000]$. Nevertheless, we notice that the shape of the curve remains unchanged with respect to varying network sizes, allowing for a *network size independent* convergence threshold to be spotted, i.e. when St_b is close to 150-200 steps. This is an important result of this study, as 1) it shows that

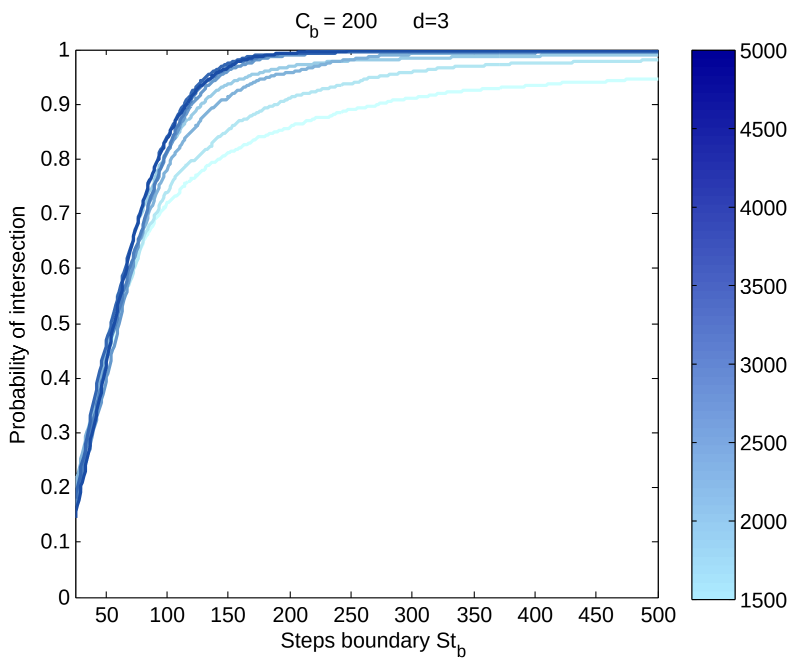


Figure 2.5: Probability of intersection as a function of steps boundary for network sizes N in $[1500, 5000]$.

the algorithm is scalable and 2) there is a good indication of a stopping parameter, to the benefit of speed and efficiency in computational power.

Concerning routing table limitations, we began all simulations with a tight memory allocation policy and gradually started to relax limitations. The range selected for C_b was $\{20, 300\}$ with a 20 steps interval and results regarding probability of intersection are presented in Figure 2.6. We notice on this graph that congestion bounding greatly influences the algorithm's efficiency in the range of $\{20, 160\}$ pushing the curve upwards, while after that point the effect on the curve shape is minimal. This is an expected result, if we consider the loss of network connectivity that can be produced by nodes becoming too quickly unavailable, sometimes resulting to big chunks of the network being isolated. Based on this graph, from now on we choose $C_b = 200$ for the evaluation of other attributes, as below this boundary we do not have a significant amount of paths that are successfully constructed.

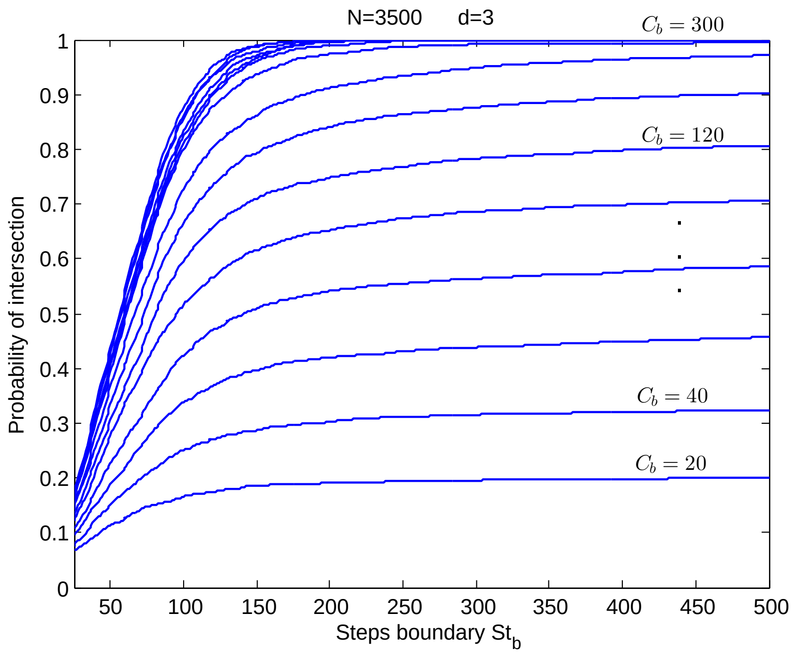


Figure 2.6: Probability of intersection as a function of steps boundary for different values of congestion boundary C_b.

It is important to notice that the St_b point where success rate converges remains unchanged during this experiment; this is an encouraging result towards determining the algorithm's configuration parameters in a possible application scenario. So far there is clear indication that a stopping point around $St_b \in \{150, 200\}$ will give optimal results independently of the network size N and the other algorithm settings.

2.6.2 Time to Intersection

The time to intersection between the publisher's and the subscriber's path is measured as the number of iterations of the algorithm until intersection occurs, which is equivalent to the total steps of the two paths, since once step is added at every iteration. In Figure 2.7, we present a statistical evaluation of time to intersection scores, for a random pair permutation of networks of varying sizes.

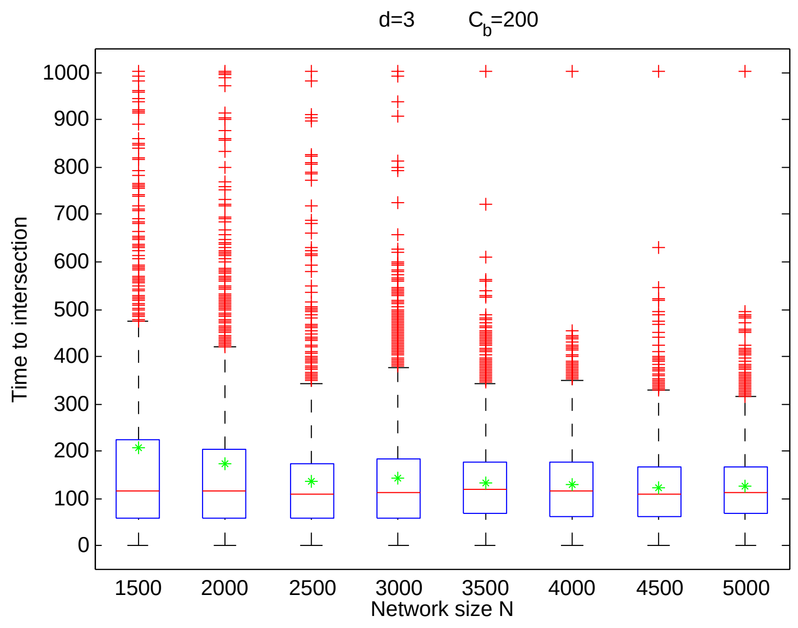


Figure 2.7: Boxplot of time to intersection results on a random network pair permutation for different network sizes N. Mean value data points are also depicted (green)

We notice that as the network grows larger there is almost no effect on median and average values and also that the interquartile range remains relatively narrow. This is a strong result suggesting the algorithm's scalability and, in addition, the converged median value (around 110 steps in this setting) is a second indication for an optimal stopping condition.

As mentioned in the algorithm design principles (sec. 2.2), it is important that the time to intersection does not depend too much on the originator nodes. For that reason, we examine all of the $N/2$ pairs in a single network permutation, to observe the relation between the distance of each pair and the resulting time to intersection. We present the results in Figure 2.8 in the form of a boxplot of time to intersection grouped by the values of publisher and subscriber hop distance. The best fit line that connects the median values reveals a small marginal impact of hop distance on time to intersection, which is another encouraging result regarding

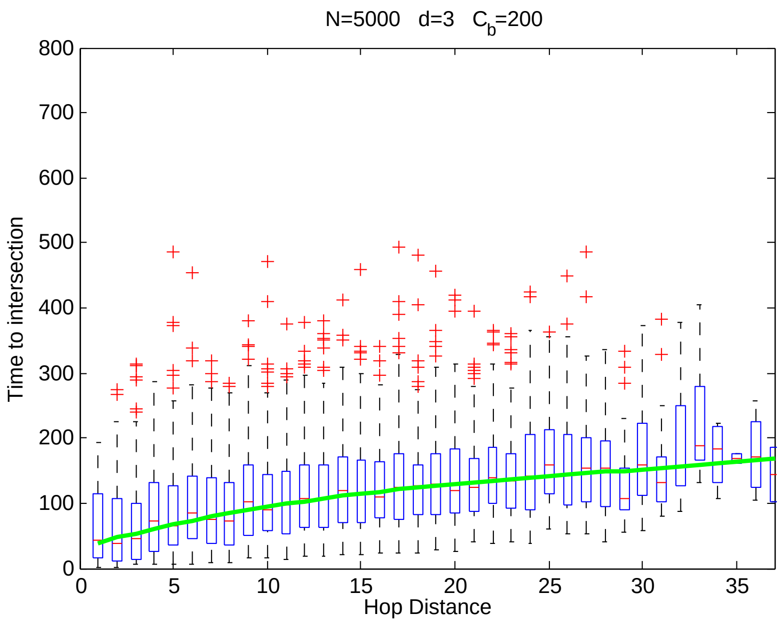


Figure 2.8: Boxplot of time to intersection results grouped by pub-sub nodes' hop distance. Results include a pair permutation of a $N = 5000$ nodes network with algorithm parameters $d = 3$ and $C_b = 200$

scalability as it implies that nodes being twice as far do not require double the time to be connected.

We have to stress though at this point that in the abovementioned graph, not all points are equally statistically significant, because of the fact that the distribution of hop distance values in a network permutation is not uniform. In fact, we demonstrate in a histogram plot (Figure 2.9) the shape of this distribution, where one can see that beyond the value of 25 there are not enough hop distance occurrences. Nevertheless, results outside the hop distance range 10-25 were similar for the multiple simulations with different settings that were performed, suggesting a stable behavior as well in this range.

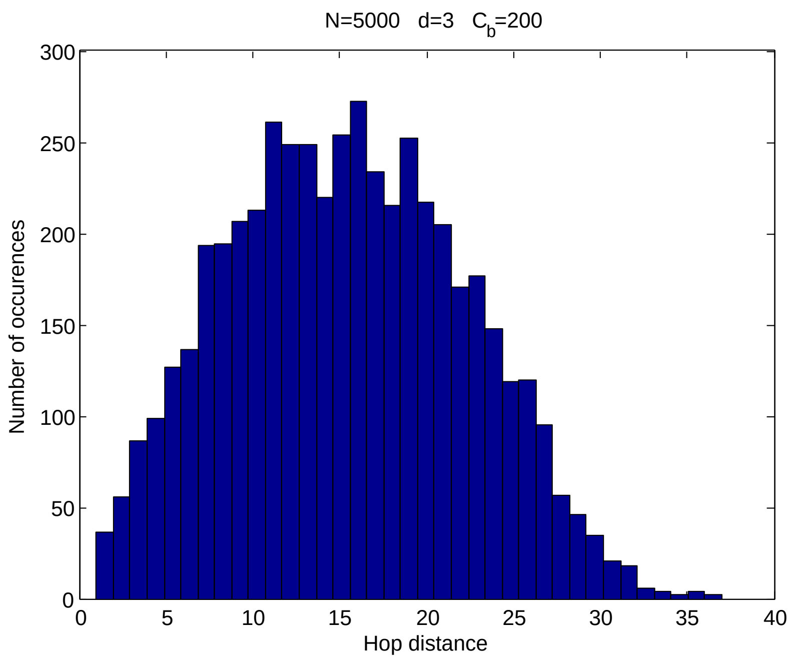


Figure 2.9: Histogram of hop distance values on the same pair permutation used for Figure 2.8

2.6.3 Network congestion

The upper congestion boundary C_b after which a node becomes unavailable, represents the constraints imposed by the limited size of RT.

For a network of size N with congestion boundary C_b we consider:

- The total available steps for all paths, or network memory:

$$M_{total} = N \cdot C_b \tag{2.3}$$

- The amount of memory needed to establish the N paths built by the algorithm is the sum of the paths' length :

$$M_{used} = \sum_{n=1}^{N} l_n \tag{2.4}$$

- The percentage of network congestion :

$$P_c = \frac{M_{used}}{M_{total}} \tag{2.5}$$

In Figure 2.10 we evaluate P_c for a network size of $N = 3500$ and a varying congestion boundary in $C_b \in [20, 300]$. We observe a low percentage of overall saturation for small values of C_b that is counter-intuitive, given the fact that a small bound normally causes more saturation, but this can be explained by the fact that for low values of C_b few paths are constructed (see Figure 2.6). Hence, in the rage of $[20 - 100]$ increasing C_b increases the number of paths and the overall congestion. After this threshold value of approximately $C_b = 100$, the relationship switches such as more relaxation causes less saturation which is the expected behaviour. We note here as well that even at its peak at $C_b = 100$ the percentage of the network that gets saturated does not surpass 20%.

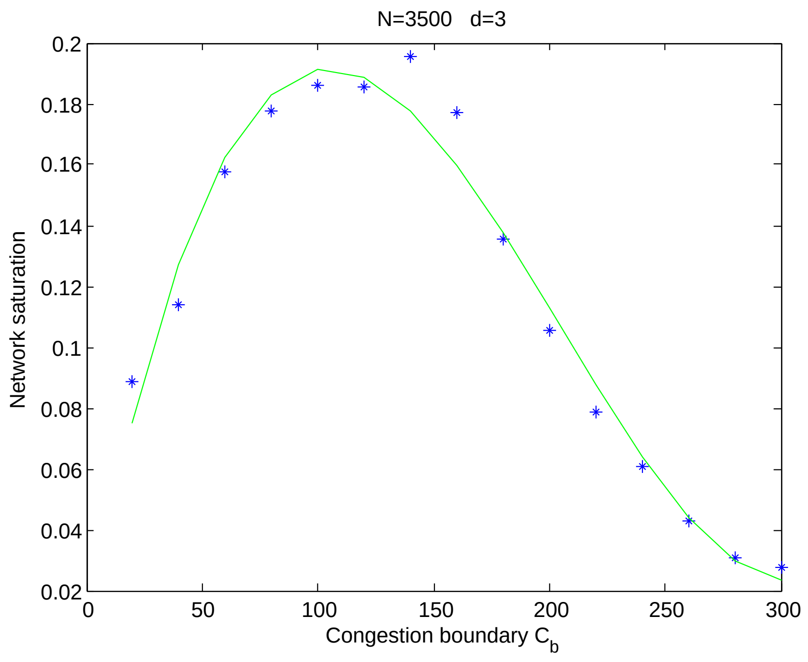


Figure 2.10: Percentage of network saturation for different values of C_b

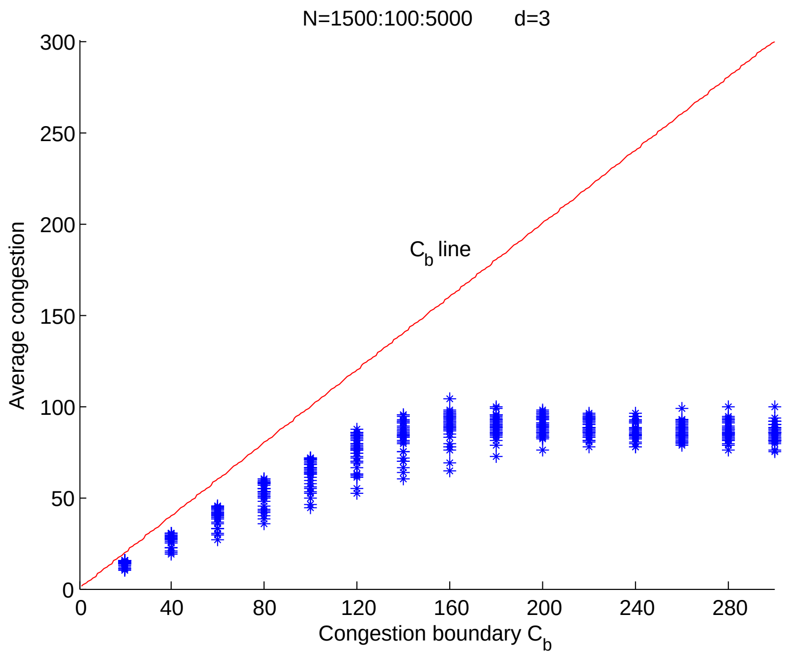


Figure 2.11: Average node congestion (blue) for networks of varying sizes and a growing C_b (red)

Another important aspect concerning congestion evaluation is the average node congestion $\bar{C}$ in the overall network after the construction of the paths. This metric gives a clear pointer on the algorithm effectiveness in terms of load balancing, which is one of the main algorithm design principles defined in sec. 2.2. For this evaluation, we calculated $\bar{C}$ for network sizes of $N \in [1500, 5000]$ with a step of 100 and different congestion boundary values $C_b \in [20, 300]$ each time. We present the results in Figure 2.11.

We can make two important observations on that figure: first, the C_b imposed is affecting the average node congestion only until a threshold value around $C_b = 160$; second, the network size doesn't seem to influence average congestion too much. The first conclusion can be easily visualized with the help of the red line indicating the value of C_b. The latter can be observed by noticing the dense distribution of the blue data points which represent the 36 different network sizes ($N \in [1500 : 100 : 5000]$), which implies that as more nodes come into play they are equally participating in the path dissemination. This is a very positive result regarding the algorithm's capacity to provide paths in a balanced way, hence not overloading a specific area like e.g. the center of the network.

2.6.3.1 Border effect

Network topology constraints can affect the algorithm's outcome, as nodes that are positioned on the edges of the network have generally reduced connectivity, which in some cases can lead to complete disconnection, and generally gives the algorithm less margin of choice for the next step of the path. In order to evaluate the impact of this border effect we employed a "sphere-ized" version of the network, where on the 2-d plane a border continuity is assumed when creating the r-disk connectivity area. This way, nodes that are positioned e.g. on the left edge can connect with the ones on the right and so on. We applied the algorithm twice for each network, once for regular and once for "sphere-ized" connectivity, and we compare the results.

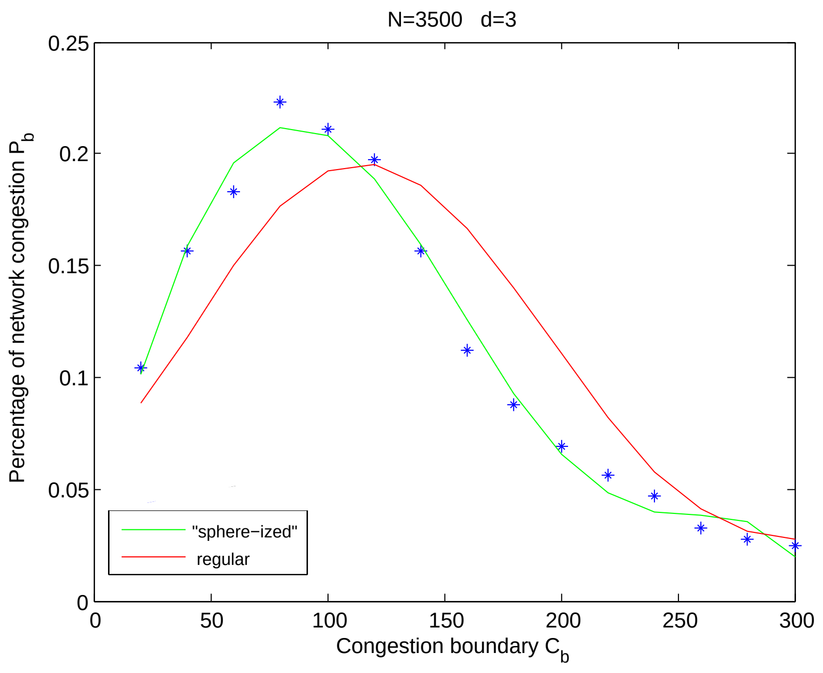


Figure 2.12: Average congestion values of a "sphere-ized" connectivity network (green) superimposed with its regular version (red) for different values of C_b.

(Figure 2.12 depicts the P_c scores of "sphere-ized" networks in superimposition with ones of regular connectivity. We notice that the change on the curve shape is minimal, which indicates that the algorithm is to some level "immune" to network topology constraints and the effects they may have.

2.6.4 Path length

Path length l refers to the length of the confirmed loop-erased path between publisher and subscriber nodes. It is important as a criteria, as paths that are too long compared to the network diameter surcharge unnecessarily the network. Since for every unit of time (one iteration of the algorithm) one step is added to one of the node's path, there is a linear relationship between path length and time to intersection. The difference is that when intersection occurs and the path is confirmed, the loops that eventually occurred during path dissemination are erased. Therefore, a way to evaluate the efficiency of the algorithm heuristic in avoiding loops is by observing the difference of average time to intersection and average path length.

We first investigate the relation between path length and network size by evaluating l in a simulation setting of a fixed $C_b = 200$ and a growing $N = [1500, 5000]$. We present the results in Figure 2.13, where one can notice a stable path length outcome throughout the experiment, at around $l = 90$ steps. This value does not diverge too much from the average time to intersection values (110) seen in Figure 2.7, which means that the applied heuristic to avoid loops is working sufficiently well. The experiment was repeated this time for a fixed $N = 3500$ and C_b in the range $[120, 300]$, to evaluate the effects of congestion bounding on path length, with the results showing a similar stable pattern (Figure 2.14). This outcome demonstrates the independence of path length with respect to network size as well as congestion boundary, a result which is inline with the algorithm's behaviour so far. Overall, the algorithm has demonstrated a stable performance when the network scales up in size, and furthermore its behaviour has been observed to remain unaffected by memory constraints beyond a relatively small threshold value.

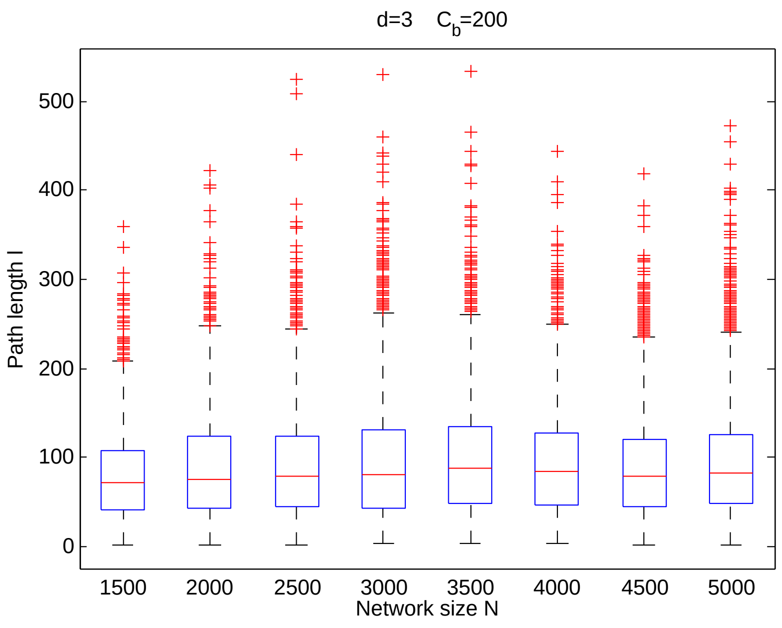


Figure 2.13: Boxplot of pathlength l for different values of network size N.

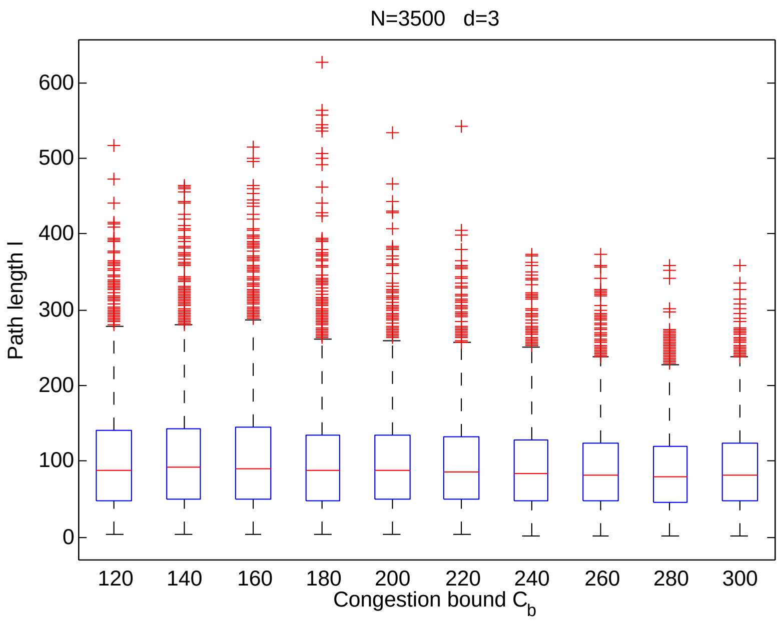


Figure 2.14: Boxplots of pathlength l results for different values of node congestion boundary C_b.

2.7 Conclusion

In this work we developed an algorithm for efficiently disseminating paths in a network where only local neighborhood knowledge is assumed. We applied the algorithm in a publish/subscribe system on top of an unstructured network, i.e. where no overlay or routing layer is yet in place, without relying on centralized broker nodes. The heuristic used to optimize path dissemination, and thus publication/subscription matching, is based on a local adaption of the Laplacian random walk. A set of extensive simulations was performed to experimentally evaluate the algorithm performance. The results show that given a reasonable computational complexity the algorithm proves to be relatively stable, with respect to changes in the network parameters. Moreover, we observe threshold values for the algorithm parameters, which as well do no depend much on the varying simulation settings.From a broader perspective, these values are chosen *locally* (node memory, number of steps) and prove to be independent of a *global* property of the communication network (f.i. network size). This observation is important since for a distributed implementation, which is what is envisioned, the global knowledge is not available.

We summarize here the key results from the experimental evaluation:

- The performance of the algorithm is sufficiently good at a depth of local knowledge equal to $d = 3$ (Figure 2.4). Further increase has marginal gains.
- Increasing the network density improves the probability of success up to a point (Figures 2.5 and 2.6).
- The time to intersection is independent of the distance of the originators and has a stable performance as network size changes (Figures 2.7 and 2.8).
- The available node memory for storing the routing table RT, which is expressed by C_b, is an important parameter for the algorithm performance.

There is a threshold effect in 2.10 which is not dependent by changes on network size 2.11 (Figures 2.10).

- The created paths are sufficiently balanced in the network (Figure 2.11).

In summary, we are tempted to claim that the algorithm scales well with the number of nodes composing the network. This claim is correct with the assumption that as nodes are added in the network, the diameter remains constant. In the case where we add nodes in the communication network while increasing the diameter, it is certain to expect an increase of the average path length, and because path length and node congestion are proportional (in the range of good performance of the algorithm) we expect again a threshold effect for C_b, i.e. there will be a value for C_b such that the algorithm works well (here it is 200), and this value will increase with the network diameter. Finally, the effect of asynchronicity was not investigated in this work, which is certain to have an effect on the algorithm performance as well.

Part III

WSN and Smart Environments

Summary

In this part of the thesis, we focus on IoT architectures and smart spaces. We delimit the challenges faced in creating robust IoT systems, which are typically linked with issues revolving around efficient energy and resource management. The demand for smart, cooperative systems, further increases the level of complexity in the above challenges, thus making the need of appropriately designed architectures even more crucial. Our work in this section is effectuated in the context of a WSN testbed, located in the TCS-Sensor lab at the University of Geneva. First, we present our work in building a location-based automation mechanism enabling smart office applications. We fully deploy the designed system and test it experimentally in various scenaria. Subsequently, we present a inter-testbed architecture, enabling the aggregation of IoT resources from various sources, as well as a 3D-modeling and visualization tool for testbed infrastructure. Finally, a method for augmenting the modeled testbeds with virtual resources is introduced, with their value assignment based on extrapolation of live as well as past data.

3 Location-based Automation for WSN-enabled Smart Spaces

3.1 Introduction

The smart spaces paradigm aims at building advanced service infrastructures that are in line with the ubiquitous computing vision: a set of smart objects that share information, interact and cooperate in order to provide a variety of digital services [33, 34]. IoT inherently plays a major part when it comes to smart spaces, as any automation function requires the capturing of its environment and its various parameters, something that is enabled via designated sensors [35, 36]. Smart homes or smart offices target to provide more convenient, comfortable, and energy efficient environments for the building occupants via installed detection and control devices, such as air conditioning and heating, ventilation, lighting, hardware, and security systems. Home automation systems have become increasingly sophisticated in recent years, with infrastructure and methods to exchange all types of appliance information and services [37].

Be it an office or a home, smart building components are typically interconnected via some IoT infrastructure, such as a Wireless Sensor Network (WSN), or provided device actuation, a Wireless Sensor and Actuator Network (WSAN) [38]. WSNs are essentially the interface between the IoT and the physical world, with their specific characteristics and challenges being the object of extensive research [39–42]. A typical WSN is composed of a set of low-power devices, small in size but still possessing an embedded CPU and a networking module, along with certain environmental sensors. In most cases there exists a central axis of communication, consisting of one or more nodes that are more powerful and/or closer to an energy source, so that they can reliably relay data to other system components. These nodes, often referred to as “gateways” can connect with the internet and eventually expose IoT resources or services to third parties [43].

Automation heavily relies in efficient data collection and processing mechanisms via a system such as a WSN, but to achieve real intelligent automation both software intelligence and hardware automation are needed. However, most of the existing products on the home/office automation market do not exhibit sufficient intelligence. A key element in achieving "true" intelligence is to be able to correctly identify the human factor in the environment. This can be abstracted at a first level to identifying things such as human presence, activity, activity status, intentions etc. To that end, location is a strong indicator towards the identification of human activity such as a person sitting, walking or running; therefore, efficient localization mechanisms are essential in smart spaces. Nevertheless, accurate indoor localization is a difficult task, so as to this day there is no established indoor positioning system achieving sufficient accuracy and which is applicable to most indoor scenarios [44].

In this work, we present an indoor location-aware smart office framework, built by integrating a WiFi-based indoor localization system with an existing WSAN testbed controlling office appliances. The system is able to track the occupants indoor location and automate accordingly the actuation of the office building appliances. The main contributions can be summarized as follows:

- We implement and deploy a WiFi fingerprinting indoor positioning system within a WSN testbed, using a smartphone application.

- We implement the automation features by enabling the indoor location-aware automation process.

- We conduct an extensive set of experiments to validate the system in complex indoor environments with long tracking paths. Results show that the indoor location-driven smart appliance automation is working well in real-time fashion.

In the following section we go over the background supporting this work, as well as discuss related work.

3.2 Background and Related Work

3.2.1 WSN and Smart Spaces

Smart spaces are becoming increasingly popular in fields such as health care [45–47], safety [48,49], education [50,51] and more. For the development of an infrastructure platform to control the electrical appliances in such environments, wireless sensor and actuator networks are the dominant technology [52–56]. WSN, rather than WiFi-based networks, are the usual choice for remote control and monitoring applications, mainly because of their low cost and reduced power consumption [57–60]. Nevertheless, the deployment of such systems is not always easy, as the heterogeneous nature of devices in smart environments demands for dedicated means in order to achieve the necessary interoperability. Moreover, the need for devices to be small in size imposes tight resource and energy constraints, which have to be carefully considered during the system architecture design [1]. Finally, seamless connectivity with mobile resources plays a vital role in smart environments, which is something that is not a default requirement in traditional WSNs and may not be easy to achieve.

Another issue that complicates the task for system designers is an overall lack of standardization in IoT enablers, regarding hardware as well as in firmware. In terms of communications standards in WSN, there are three dominant solutions:

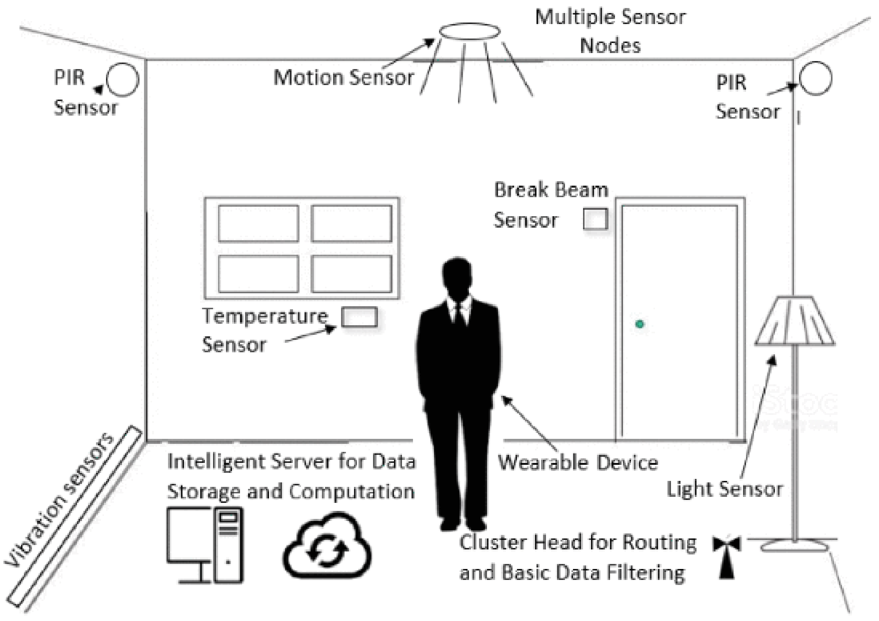


Figure 3.1: A typical smart space [1].

ZigBee [61], Bluetooth [62] and 6LoWPAN [63]. ZigBee is a low-power specification protocol which possesses short communication range (up to 10 meters) and limited data transfer rate (up to 250 kbps), although its lightweight design keeps its power consumption very low. Bluetooth was designed for data exchange between mobile devices over short distances. It can scan for devices in up to 100 meters range and transfer data with a rate up to 3 Mbps, keeping the power consumption still at low levels. Finally, 6LoWPAN is an adaptation of the IP protocol for low-power devices, allowing for IPv6 packet to be sent and received over sensor networks. Its communication range goes up to 90 meters and its power consumption is medium.

In recent years, a shift to IPv6-based architectures can be observed in WSAN [64–67], as they simplify integration of sensor nodes into the globally prevalent IP-based networks, thus facilitating cross-platform communication. The emergence of embedded operating systems, tailored for low-power devices, such as the popular TinyOS [68] and Contiki [69], have significantly facilitated management at the application layer in modern WSNs. Moreover, the Constrained Application Protocol (CoAP) [70], a CRUD-based protocol for WSN, has enabled the implementation of

RESTful architectures following HTTP and the web service paradigm [71]. The Syndesi testbed [72], at which our work is deployed, combines WSANs with different communication technologies such as Near Field Communication (NFC), Bluetooth and 6LoWPAN, along with an electrical interface to control office appliances, in order to create personalized smart environments. Syndesi architecture and components are detailed in a following section.

3.2.2 Indoor localization

Determining people's location is an essential requirement for building smart office or smart home applications of true intelligence. Based on satellite signals, The Global Positioning System (GPS) can provide accurate positioning for outdoor environments, although it fails to deliver respective results in indoor settings due to poor line of sight. Various techniques have been developed to overcome this issue, the most widely used employing fingerprinting and ranging methods which estimate the location of an object by measuring its distance from some local anchor points (beacons) [73, 74]. The methods for extracting this distance vary, the principal ones being the time of flight (ToF) [75], time of arrival (ToA) [76] and received signal strength indicator (RSSI) [77]. ToA and ToF based methods use packet transmission to estimate the time lags between wireless devices, achieving high precision, but with a steep price on hardware costs and implementation difficulties [78]. On the other hand, RSSI-based techniques highly depend on the environment and surrounding structure, which overall limits significantly their accuracy. Nevertheless, given the trade-off between accuracy and applicability at system design, RSSI being a very cost-effective method is currently the most widely used technique for indoor positioning systems.

Another factor which influences the accuracy of an indoor positioning system (IPS) is the choice of transmission technology. Wi-Fi, Bluetooth, Radio Frequency Identification (RFID), Ultra-wideband (UWB) and magnetic field distortion are the dominant technologies used, with Wi-Fi and Bluetooth being the most frequent choice [79]. Studies [Zhao et al., 2014] comparing Wi-Fi and Bluetooth based positioning indicate Bluetooth as a better candidate due to its lower transmission power, higher scan rate and more frequent channel hopping [78, 80]. That being said, WiFi has the advantage that necessary infrastructure may be already in place as WiFi APs are provisioned in most buildings top provide wireless internet access.

Beyond signal processing, another approach for enhancing indoor positioning systems is through inertial measurement units (IMUs) [79]. The sensors utilized as IMUs are typically among accelerometers, gyroscopes, magnetometers and barometers. The most common approach that employs IMU sensors for navigation is pedestrian dead reckoning (PDR) [81], in which the accelerometer is used to estimate the displacement of a moving object, while the gyroscope and magnetometer compute the direction headed. IMU-assisted positioning is often combined with RSSI fingerprinting using a filtering mechanism, so as to eliminate each others drawbacks (f.i. PDR requires the initial position of the object) [82–84]. The most popular filtration methods used for this merge are the kalman filter and the particle filter, as well as their variants [85, 86]. In the following section we detail the algorithm used for localization in this work, which is based on WiFi RSSI fingerprinting.

3.3 Indoor Location for Smart Environments

This section details the design of the proposed location-aware smart environment framework. Initially, we present the overall system architecture and then we go over each of its components while discussing the integration challenges. Our framework comprises of a core WSN testbed, *Syndesi*, upgraded with a localization engine enabling smart automation via a smartphone application. We refer to the upgraded WSN with location-based automation as *Smart Syndesi*.

3.3.1 Overall Architecture

Smart Syndesi, as shown in Figure 3.2, is a system comprised of heterogeneous hardware devices managed in a unified way with intelligent software engines. It has three main components, namely the *Syndesi* wireless sensor and actuator network testbed, an indoor localization and navigation engine, and a human-centric actuation automation module.

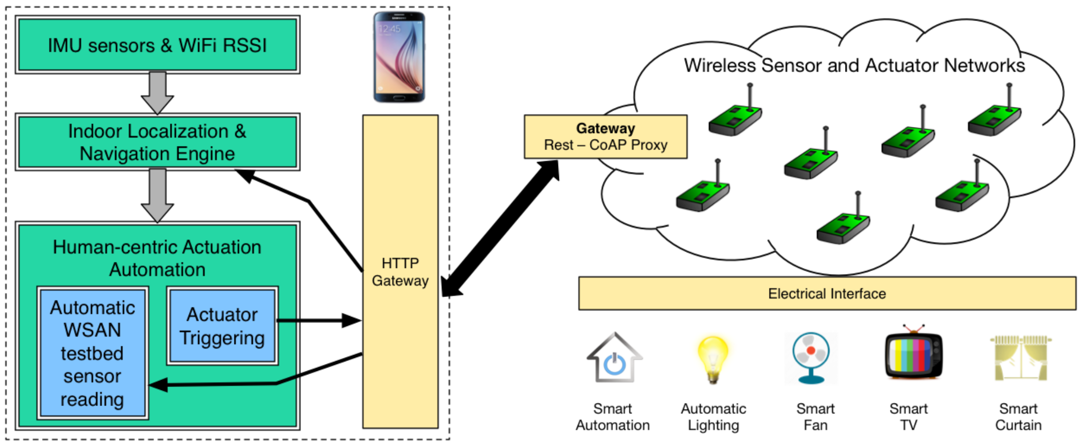


Figure 3.2: Smart Syndesi system architecture.

The *Syndesi* testbed is responsible for creating and managing personalized smart environments using WSANs. It exposes its resources in an abstracted fashion via a designated gateway, while providing control of electronic appliances that are relayed to the WSAN using an electrical interface. The indoor localization and navigation engine is built in an Android application and can estimate the location of the users in real-time fashion. Finally the human-centric actuation automation module resides in a server which is linked with the mobile application and *Syndesi* via the WSAN gateway, enabling the activation of correlated actuators based on the estimated user locations. In the following, we describe the functions of each module in detail.

3.3.2 Syndesi Testbed

The Syndesi testbed is a system and a platform comprised of heterogenous devices, sensors and resources focusing on the Internet of Things. Through multiple entry points, a large number of heterogeneous devices are able to interconnect with the testbed and provide their resources. It integrates sensor networks, electrical appliances, actuators and gateways via various communication protocols and technologies such as Near-Field Communication (NFC), Bluetooth, ZigBee, 802.15.4, 6LoWPAN etc. The scope of Syndesi is not only to monitor environmental parameters but also to serve as system-as-a-service to its users. Its architecture is designed with scalability and interoperability in mind, which allows even mobile resources and data to be aggregated in it.To that end, the Syndesi testbed follows a RESTful architectural approach providing interoperability between its resources, devices, services and the internet. Benefiting from Syndesi's REST-enabled services, external requesting systems are able to access and manipulate textual representations of its resources that are exposed to the Web, using a uniform set of stateless operations. Finally, via a designate mobile application, a mobile crowd sourcing component is provided, via which smartphone users can contribute with data collected from their smartphone sensors directly into the server's database.

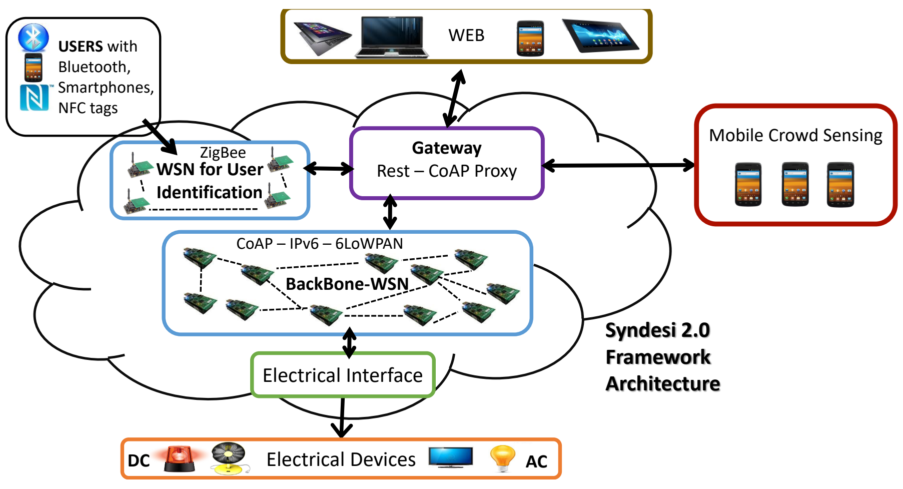


Figure 3.3: Syndesi testbed architecture

The overall architecture of Syndesi is shown in Figure 3.3. The core WSAN is comprised by 28 TelosB [36] and Zolertia Z1 [?] sensor motes which are connected with the existing electrical and electronic office devices, such as lights, fans, electric curtains etc., via solid state relays. The resource management functionalities are implemented on a Linux server which resides behind the WSN gateway. That way, it can serve as a connection point for all the system components, such as the WSAN, the mobile crowdsensing smartphone application, the web etc. Every service or resource is provided as a web service, in the form of a RESTful API deployed also in the gateway server, and linked to the web through a proxy service. Moreover, an SQL database is hosted on the same server where sensor data can be pushed via corresponding APIs. As REST architecture implies, the API calls have the simple form of a URI utilizing GET or POST methods. Syndesi components are deployed in 4 office rooms in the University of Geneva premises where 7 people currently work. In figure 3.4 we present a 3D-model of one of the offices. The red marks indicate the electrical devices that are connected to Syndesi and thus capable of being triggered via the designated APIs. Most of devices such as the fans, lights and coffee machines support switching on/off while the electric curtain motor is actuated up/down. Some sensor motes (green marks) are placed on the wall for purely monitoring purposes. Using those motes, an automated polling scheme is set-up since September 2016 which collects environmental data, such as temperature and illuminance, and stores it on the server database following a predefined temporal pattern.

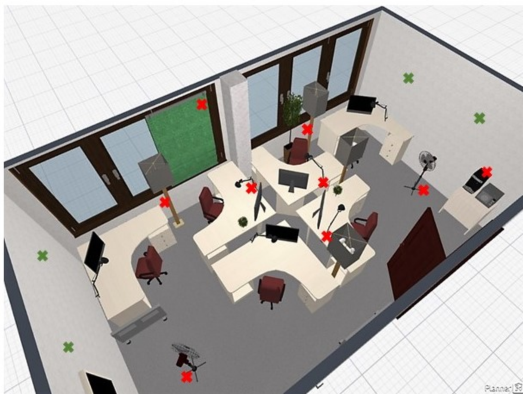


Figure 3.4: 3D representation of office environment. Green marks are sensors and red marks are actuators.

3.3.3 Indoor Localization Engine

To track people in indoor environments, we have designed an indoor positioning system to support real-time indoor localization and navigation. Our approach [87] is able to provide high accuracy by fusing smartphone on-board IMU sensor readings, WiFi RSSI from local APs, together with floor plan information using a particle filter. The tracking algorithm is designed to run exclusively on a smartphone, which would prevent additional hardware requirements that would with increase the complexity in an application scenario. Figure 3.5 depicts the system architecture of the indoor positioning system, which contains four components as follows:

Inertial Measurement Unit

In order to estimate the user displacement, we use two sensors, the accelerometer, and the geomagnetic field sensor. The displacement of the user at time t is defined by the motion vector $M_{v_t} = [\theta_t, \ell_t]$, where θ_t is heading orientation and ℓ_t is stride length at time t. Time t is the instant that the user executes a new step. Therefore, step recognition and heading orientation methods are implemented in this component. The step recognition method is developed by parsing acceleration readings, whereas the heading orientation is determined by a digital compass derived from the geomagnetic field and accelerometer sensors. Despite the fact that stride length can vary along the trajectory, we assume that ℓ is a constant value to simplify the task at hand.

Floor Plan Component

Information about the area of interest is used to further improve the tracking accuracy. This component defines the physical constraints imposed by the environment of the area of interest. Therefore, zones in the floor plan where for instance it is not possible for a user to walk, e.g. walls, furniture etc, are marked as such and the algorithm then utilizes this information.

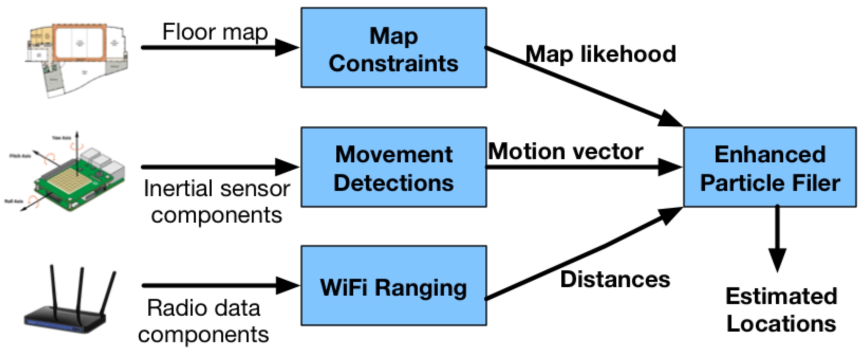


Figure 3.5: Indoor positioning system combing i) WiFi RSSI ii) IMU sensor readings and iii) floor map constraints using a particle filter.

Radio Information Component

Radio information needs to be converted to range values. In order to achieve high ranging accuracy we adopt the Non-Linear Regression (NLR) model presented in [88]. The NLR model is defined as follows:

$$d_{j,t} = \alpha_j * e^{RSS_{j,t} * \beta_j}. \tag{3.1}$$

$d_{j,t}$ is the distance between the target object and the jth AP at the instant t. Both α_j and β_j are environmental variables defined for the jth AP. $RSSI_{j,t}$ is the signal power measured from the jth AP at time t.

Data Fusion Component

We use a particle filter to fuse IMU, WiFi, and floor plan information in order to achieve real-time tracking in indoor environments. In our approach, an additional re-sampling method is incorporated in order to further mitigate the errors caused by off-the-shelf WiFi sensors embedded on commodity smartphones. Therefore, the state vector at time t is defined as follows:

$$X_t = [x_t, y_t, \theta_t, \ell_t], \tag{3.2}$$

(x_t, y_t) are the Cartesian coordinates of the target object, θ_t is the heading orientation and ℓ_t is the stride length at time t.
The prediction function can be written as:

$$X_t = F \cdot X_{t-1} + \eta, \tag{3.3}$$

where

$$F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \eta = \begin{pmatrix} \ell * cos(\theta) & 0 \\ 0 & \ell * sin(\theta) \\ \theta & 0 \\ 0 & \ell \end{pmatrix}$$

$$\theta = \theta^{'} + \varepsilon^{'}$$
$$\ell = \ell^{'} + \varepsilon^{''}$$

Heading orientation and stride length are assumed to interfere by zero-mean Gaussian random noises. Therefore, $\varepsilon^{'}$ and $\varepsilon^{''}$ are the errors introduced in the calculation process.

The above positioning system has been tested in [87] and was able to run exclusively on a smartphone, as was designed. That being said, it may be not necessary to implement the full tracking algorithm for every application; some applications might need sub-meter level accuracy, while for others meter level accuracy could suffice. In the next section we examine the requirements of integration in the context of Syndesi testbed and propose some necessary adaptations.

Integration with Syndesi Testbed

Before the integration of a localization mechanism, the Syndesi testbed included a mobile crowdsensing application for users to collect sensing data using their smartphones. For every new user, this application asked for an office room corresponding to their working location, to be used as a location stamp on their sensed data. The motivation in this work was to integrate the localization module described in the previous section into the existing mobile crowdsensing application. That way, crowdsensed data could automatically be location-stamped, and moreover an automation module could be built based on the location as well as the sensed data.

Given the increased resource demands of the full tracking mechanism, and taking also into account the fact that a sensing unit and an automation unit will also consume smartphone resources, we decided to adapt the localization algorithm to a more lightweight version, with an inevitable trade-off in accuracy. Nevertheless, we regard a room level localization accuracy to be sufficient for smart office automation scenaria and therefore propose an approach that is able to provide that with high confidence as efficiently as possible. In this regard, we choose to keep only the radio signal component of the IPS detailed before (see Fig. 3.5), in a simple WiFi fingerprint room recognition approach based on the WiFi RSSI transmitted by nearby WiFi access points. The key idea is to combine the signal strength from multiple access points in order to build WiFi fingerprints for supervised learning.

The proposed approach consists of two phases: the off-line training phase and the on-line localization phase. The training phase is intended to build the fingerprint database, which consists of vectors of WiFi RSSI collected from the nearby WiFi access points. During the training phase, each survey room is characterized by a set of vectors of WiFi RSSI readings. During the on-line localization phase, the predicted room likelihood is calculated based on the previously collected information and the current WiFi RSSI vector. Thus, room recognition can be handled by searching for the closest match of the test data in feature space. We implement two simple, yet proven to be efficient, classification algorithms for this purpose: the K-Nearest Neighbor (KNN) [89] and the Support Vector Machine (SVM) [90] classifiers. KNN and SVM are some of the simplest classification algorithms available for supervised learning. Every time a localization task is launched in the mobile application, both classifiers are employed and in case of a classification mismatch, the process is repeated with a newly collected WiFi RSSI vector.

3.3.4 Mobile Application

The mobile application has been developed in Android Studio and IntelliJ IDEA using the Android SDK with API level 21. It supports devices from API level 15 and above, which at the time of writing encompasses over 99.1% of Android smartphone users [91]. Figure 3.6 depicts the high level application architecture, which is comprised of three discrete units, a sensing unit, a localization unit and a control unit. The *sensing unit* is responsible for querying the smartphone sensors for values, and corresponds to the previous version of the app which was focused on mobile crowdsensing. The *localization unit* is responsible for estimating the smartphone location with the mechanism detailed in the previous section. Finally, the *control unit* coordinates the various actuation/automation tasks as well as the resource and power management. The application is constructed around the Model-View-Controller (MVC) pattern [92] in which users, sensor data and nodes are models, views are handled by Android activities and controllers are using services to run in the background. The application follows the Separation of Concerns (SoC) design principles [93] which facilitate the development as well as the maintenance. All data that is transmitted to the server are first converted to the JSON format which is widely used and favors interoperability.

3.3.4.1 Sensing unit

Once a user grants the necessary permission to access the device hardware, the application discovers the available sensors in the device and establishes a monitoring

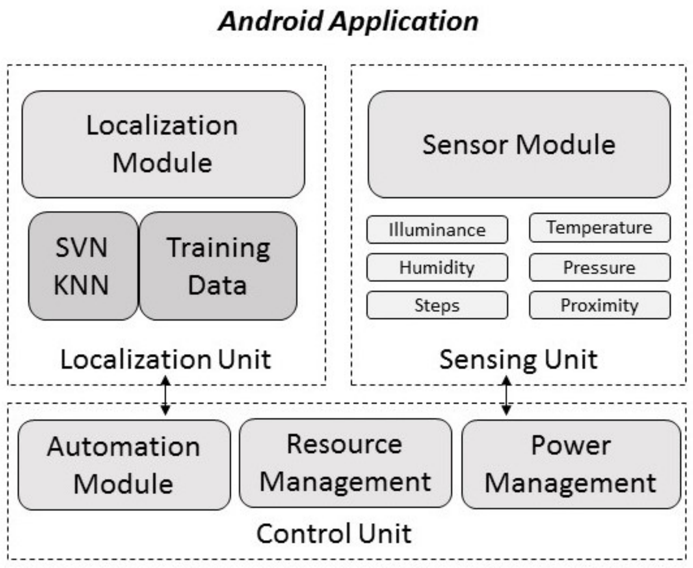


Figure 3.6: Integrated Android application architecture.

scheme. The polling rate can be manually set in the settings, the default value being one measurement per minute. After each new sensing event, the raw data is converted to JSON after being stamped with the user's id as well as the location which is provided by the localization unit. Then, a network controller communicates with the RESTful service provided by the Syndesi server and transmits the data to the server via an HTTP call. This controller can be easily provisioned to create subclasses in order to connect to other types of servers, making the app easily adaptable to different environments. The HTTP requests are made using Volley, an HTTP library made for Android that permits more efficient and faster networking by using caching, multiple concurrent connections and an automatic scheduling for network requests. All tasks performed by the sensing unit are run as services that can run also in the background, therefore allowing for continuous data collection even during the smartphone's idle periods.

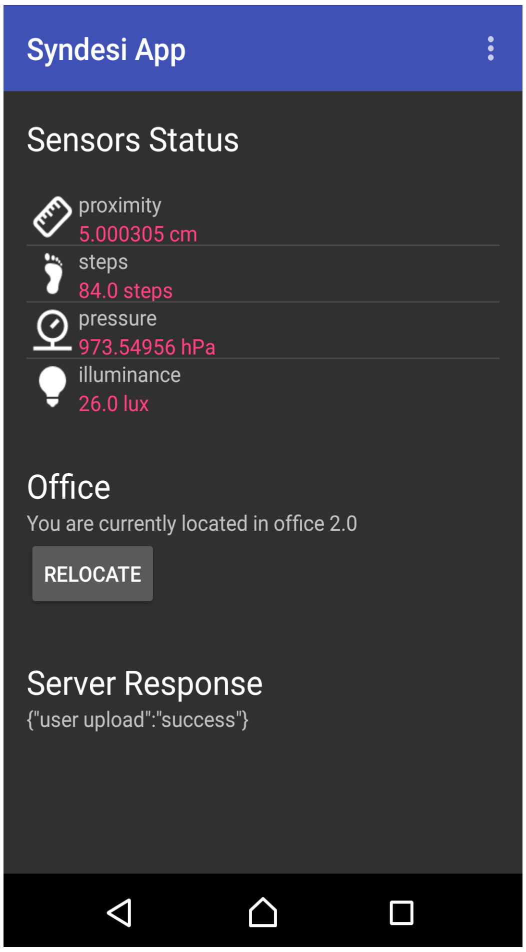
3.3.4.2 Localization unit

The localization unit is where the localization tasks are implemented. As mentioned in section 3.3.3 the full tracking mechanism was evaluated as too power-consuming to be used in this mobile app, so we adopted a lightweight version of the same algorithm to achieve accurate room-level localization. The lightweight version uses only the radio information component to derive the user room, in a temporal interval defined by the user. When a localization task is launched, the app scans for surrounding WiFi signals and forms a new WiFi RSSI vector which is then fed on the SVM and KNN classifiers. The classifiers are trained every time the app is initially launched, providing that way the possibility of training on different data without the need of re-installation of the whole application. Similarly to the sensing unit, all localization tasks are run as services that can be run on the background. The classifiers used for the localization come an Android adaptation of OpenCV [94], an open source computer vision and machine learning library.

3.3.4.3 Control unit

The control unit provides a control interface for the user to manually trigger the testbed appliances. This is enabled via a periodic polling to the gateway for the list of sensor/actuator nodes registered in the WSAN. The nodes are parsed from the JSON response and displayed on the application screen, where the user can toggle a actuator node by clicking on its icon. The click triggers the network controller which sends a mediate (HTTP GET) request to the server with appropriate parameters and the corresponding appliance is actuated. Besides the manual triggering of the actuators, the control unit includes an automated environment control module which uses the tracked indoor location to activate smart management within the Syndesi framework. If a user enables that option on the settings, when a change of location is detected it triggers a process that switches on the appropriate appliances, such as lights that are in the same room as the device, switching off the ones in the previous room if no one else is currently located inside. The user can also set in the environment control configuration desired values for ambient temperature and illuminance, which when compared to the live measured values will actuate an office fan, turn off some lights or pull up/down a curtain in corresponding cases.

Figure 3.7 shows a screenshot of the application home screen, where the latest sensor values are displayed as well as the current location and the server status. Figure 3.8 displays the app settings page, where the different settings of the application can be configured, such as the polling rates and the various permissions.



Syndesi App

Sensors Status

proximity
5.000305 cm

steps
84.0 steps

pressure
973.54956 hPa

illuminance
26.0 lux

Office

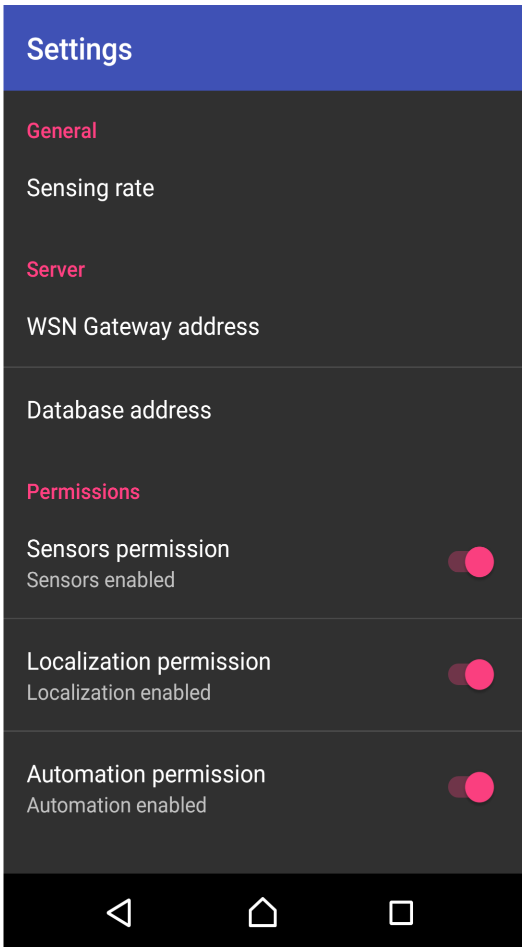
You are currently located in office 2.0

RELOCATE

Server Response

{"user upload":"success"}

Figure 3.7: App home page.



Settings

General

Sensing rate

Server

WSN Gateway address

Database address

Permissions

Sensors permission
Sensors enabled

Localization permission
Localization enabled

Automation permission
Automation enabled

Figure 3.8: Application settings.

3.3.5 Integration challenges

Machine learning on resource constrained devices

The first challenge when integrating the indoor localization algorithm in the mobile application was the inclusion of the different classification algorithms such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM). Presently, there does not exist separate pure machine learning libraries developed for Android, so we decided to utilize the OpenCV library, even though it was initially designed for image processing tasks. This library is open source and has already been implemented for Android, the downside being that it has to be installed separately, something our app prompts the user to do if it is not already installed. A different approach would be to move the localization computations on the server side, something that would greatly increase the freedom in algorithm choice, as resource considerations would not be an issue anymore. In that case though, the seamless connectivity between phone and server needs to be ensured as the additional delay from the round trip (send input RSSI/ get localization result) could compromise the real-time feature of the application.

Battery consumption

The initial version of the app was configured to collect data every 3 minutes and in a full working day (8 hours) we measured an average of 3% battery consumption. Although, as new features continued to be added, e.g. faster polling rates, computationally expensive localization tasks, node management etc., it became significantly more power demanding. To deal with this issue, we introduced a new power management feature, where the polling rates as well as the localization WiFi sampling intervals are bound to the smartphone's current state and battery level. The user can still choose a desired polling rate in the settings but when the phone battery reaches levels below 50%, the polling rate is reduced by half, and furthermore if the battery life goes below 20% the sensing and localization functions are both disabled. The power efficiency can be further improved by batching data that are not critical to automation before sending them to the server, as well as by activating the localization only when a user displacement is detected.

3.4 System Deployment and Evaluation

In this section, we explain in detail how the system is deployed, in an indoor office area at the University of Geneva, as well as the methodology used for its evaluation.

3.4.1 System Deployment

The system was deployed at the TCS-Sensor Lab in the Computer Science department of University of Geneva with an office area of nearly 260 m^2. In Figure 3.9 a floor map of the offices is depicted, together with the system actors as well as the evaluation paths. In order to cover the area of the 4 office rooms and the in-between corridor with WiFi signals that are sufficient to feed the localization algorithm, 5 WiFi access points were strategically deployed. Each office possesses multiple sensors (e.g., humidity sensor, illuminance sensor, etc) as well as actuators paired with electrical appliances (e.g., desk fans, desk lights, and electrical curtains). Each actuator is linked to a sensor mote via a solid state relay, which in addition to the sensor mote it is connected to the 220V power supply and the electrical appliance. When a location-based actuation is identified by the system (in the app), the actuator mode outputs a 3V voltage to the relay. The current generated by this voltage enables the relay's underlying circuit and the correlated appliance is switched on. In a similar manner, devices are switched off, or actuated up or down (electric curtain). During the experimental evaluation, a person walked through all office areas following designated paths, holding a smartphone with the *Smart Syndesi* app installed and enabled.

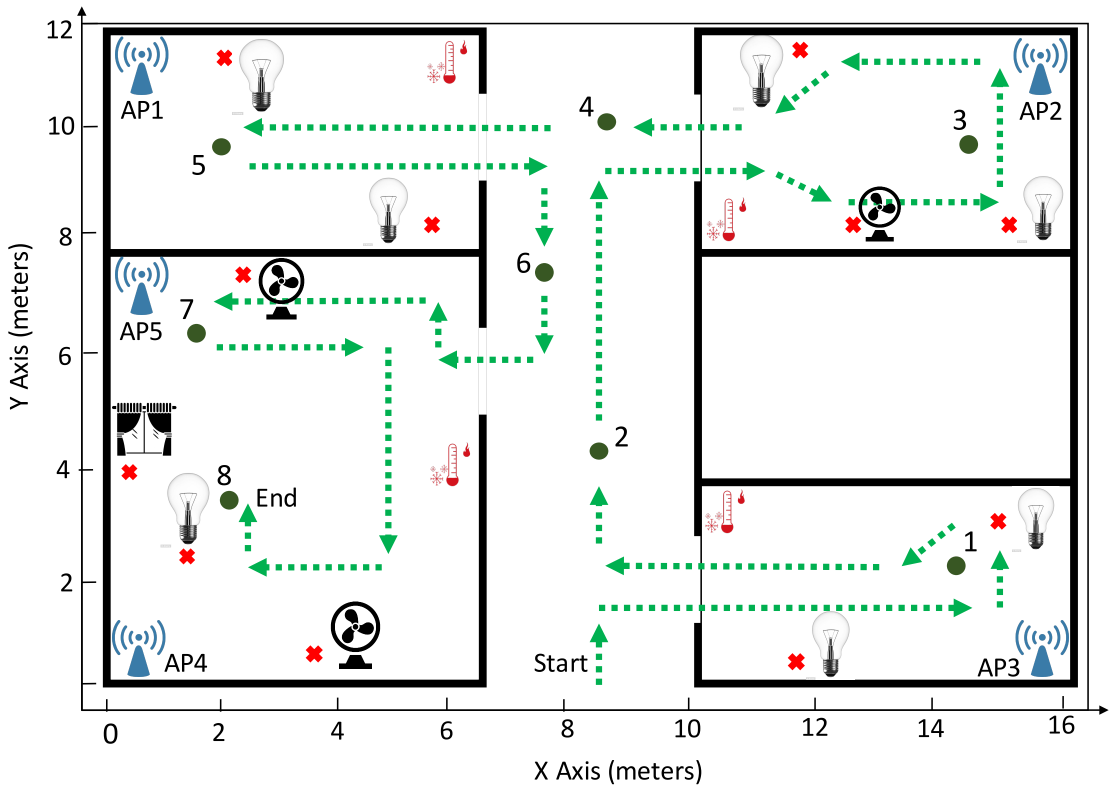
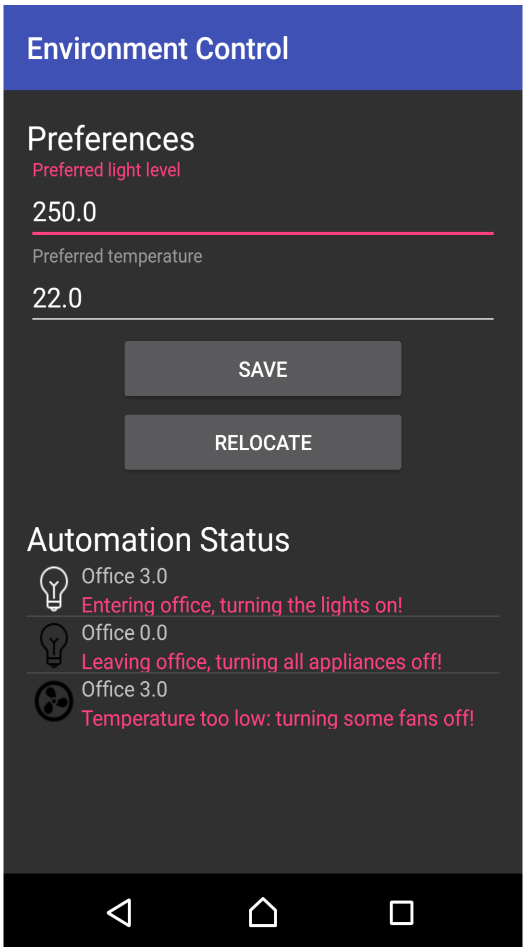


Figure 3.9: Experiment scenario.

3.4.2 Evaluation

The experimental evaluation of the deployed system can be viewed as to include two aspects: a functional one, and a non-functional one. The functional evaluation consists of validating the prototype functionalities, while the non-functional refers to the evaluation of the indoor localization accuracy.

The app interface of "Node Manager" (Figure 3.11) displays the actuators that are in the current room and their status. Moreover, as shown in Figure 3.10, the "environment control" feature allows the user to set their preferences regarding their office's environmental conditions. Based on those values and while monitoring current readings, the app triggers the appropriate appliances and displaying any action taken in its interface (bottom of environment control tab Fig. 3.10). Therefore, for the functional evaluation, every time a person holding the


Environment Control

Preferences

Preferred light level

250.0

Preferred temperature

22.0

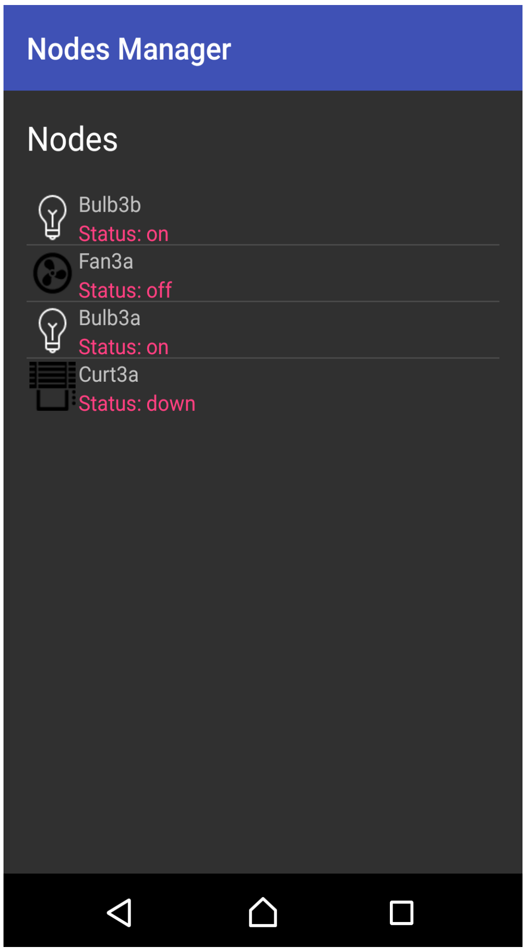
Automation Status

Office 3.0
Entering office, turning the lights on!

Office 0.0
Leaving office, turning all appliances off!

Office 3.0
Temperature too low: turning some fans off!

Figure 3.10: Automation settings.


Nodes Manager

Nodes

Bulb3b
Status: on

Fan3a
Status: off

Bulb3a
Status: on

Curt3a
Status: down

Figure 3.11: Nodes Manager.

smartphone changes room we measure if the actuators of the corresponding office are correctly triggered based on the preferences.

For both the evaluation of the accuracy of the room recognition, as well as the automated environment control, an experiment scenario was defined. The experiment consists of a person holding the smartphone and walking through the office areas following a predefined path, shown as green dashed line in Figure 3.9. During the designated walk, the person holds the smartphone and enters office rooms one by one, going over a set of predefined "check points" on the ground (shown as number 1-6 in Figure 3.9). At each check point, the "relocate" function is manually triggered manually on the app to launch a new localization attempt.

Model	Specifications
Samsung Note 3	Android 5.0; Quad-core 2.3 GHz Krait 400; Wi-Fi 802.11 a/b/g/n/ac; 3GB RAM
Sony Xperia Z5	Android 7.0; Quad-core 2.0 GHz; Wi-Fi 802.11 a/b/g/n/ac; 3GB RAM
LG Nexus 5X	Android 7.1.2; Hexa-core 1.8 GHz Cortex-A53; Wi-Fi 802.11 a/b/g/n/ac; 2GB RAM

Table 3.1: Smartphone model specifications

We note at this point that for the regular version of the Smart Syndesi app, an automatic triggering at every detected user displacement is provisioned, but at this point we chose to focus on the accuracy of the localization, as well as the correct appliance automation. At the time of the experiment, the mobile app was configured to have the environment control function enabled, therefore the expected outcome was the enabling/disabling of the corresponding electrical appliances as the location changes. As measured WiFi received signal strength can depend on the smartphone hardware components, we used 3 smartphone models from different manufacturers to see if there will be a divergence on the results. Their detailed characteristics are shown in Table **??**.

The experiments were run 20 times for each phone and for every checkpoint we gathered metrics on the correctness of room recognition and appliance actuation. We present the results in Figure 3.12. The first observation is that the overall accuracy of the room recognition was far greater than 90%, which validates the efficiency of the deployed localization system. In particular, we notice that check points 1,3,5,7 and 8 which are all located inside the 4 offices achieved perfect score. This is an important result since when it comes to localization misses, a corridor miss is far less critical since it is basically a transitive state. Regarding the actuation of office appliances, measuring only when the office room was correctly detected, the system demonstrated robustness and reliability as there were zero failures in the sequence of steps taken by phone-server-WSAN to enable the automation, as can be seen in Figure 3.12.

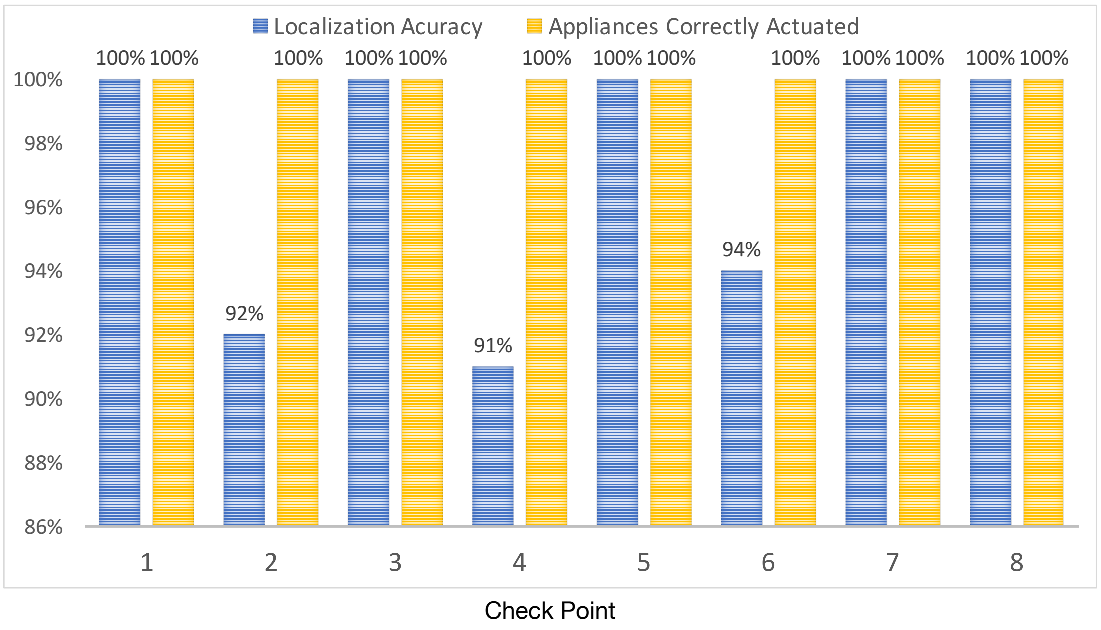


Figure 3.12: Evaluation results.

3.5 Conclusion

In this work, we designed, deployed and experimentally evaluated an indoor location-aware smart environment system. The system consists of a wireless sensor and actuator network for electrical appliance management, interconnected via a gateway with a smartphone application equipped with an indoor localization module. The mobile app enables the collection of environmental data utilizing the smartphone built-in sensors, which are sent back after being location-stamped to the server via the gateway. Moreover, thanks to the accurate indoor localization module, the system is able to identify people's indoor locations in real-time and enable automation of office appliances. We implemented the system prototype, and deployed it in an indoor environment of 4 office rooms in the TCS-Sensor lab of at the University of Geneva. In order to validate the system functionalities and performance we performed a set of experiments using different smartphones models. Evaluation results show that the estimated indoor locations are of high accuracy, and the automation of office appliances is robust and can function in real-time.

4 Modelling of WSN Testbeds and Augmentation with Virtual Resources

4.1 Introduction

In applied research fields such as networking and computing, testbeds and other experimental facilities are essential for investigating novel methodologies. Prototype architectures and protocols need to be deployed and evaluated in a controlled environment which is easily re-configurable, and where selected parameters can be monitored in an effortless way. Furthermore, the support for agile architectures should be coupled with additional tools for gathering or processing meta-data in the context of an experiment (e.g. calculating different evaluation metrics or analyzing execution logs after an experiment run). Simulation tools offer a powerful alternative to experimental evaluation, but in fields of applied research such as IoT some physical experiments are necessary to support a hypothesis in addition to simulations.

In the field of WSN and IoT, significant efforts have been put forward in establishing such facilities, with notable examples being Fed4Fire [95] and IoT-Lab [96]. However, no matter the volume of invested resources, the infrastructure and services provided by testbeds will always come with some drawbacks. For instance, by nature, all testbed facilities are designed with a specific area of interest in mind, be it IoT applications, Machine-to-Machine (M2M) communication protocols or else. This leads to a certain "overfitting" in the architecture and the provisioned tools, which therefore imposes some limitations to the range of supported experiments. Another constraint is that the number of available resources at at a given time depends on the current user demand, as in most cases testbed infrastructure cannot support simultaneous experiments without jeopardizing the outcome value. Expanding the experimental facility is a possible solution to the above problem but, whether that refers to an increase of its size or the number of simultaneous experiments it supports, or its availability, scaling up always entails considerable expenses and a large amount of effort.

With the aforementioned problems in mind, we propose in this work a different approach than the one of traditional testbed facilities in the field of IoT; we introduce a hybrid testbed type, comprising of a physical component and a virtual one. The physical component contains regular IoT resources (f.i. IPv6-enabled sensor motes forming a WSN) that are physically deployed in the facility premises, and following traditional testbed architecture, are controlled by a back-end system that monitors and orchestrates their operation. The second component, which is where the novelty of our proposal lies, consists of virtual resources that are operating based on a representation of the physical one. This happens through simulation mechanisms capturing the facility's physical space (e.g. line of sight, light distribution) as well as the physical resources' functionalities (e.g. networking operations). In order for the virtual component to qualitatively as well as quantitatively augment the physical one, the calculating of sensor value at virtual resources needs to be accurate. To that end, we utilize a combination of spatial extrapolation techniques with associative analysis on past data, which is evaluated using K-fold cross validation. Indicatevely, results such as Mean Absolute Percentage Error (MAPE) at levels below 7% - are encouraging and show that high accuracy can be achieved.

In order to fuse the two components in an intuitive way, we employ a 3D representation model of the physical testbed, equipped with an interactive Graphical User Interface (GUI). The purpose of the GUI is twofold; first, using the GUI, a researcher can get a better "feel" of the testbed's physical space, including parameters such as topology of rooms and volume of objects, which can help them capture things that may be less obvious in a traditional map. Second and most importantly, thanks to the provided interactive mechanism, users are able to spawn virtual resources within the digital model with a click of a button, after which a virtual resource is created to the corresponding location. In the modelled testbed, the virtual resources are bound to the same physical limitations as the physical ones (e.g. a virtual sensor cannot be spawned midair), the end goal being a seamless integration of virtual and physical to the point of being perceived as a single unity.

We build the first prototype modelled testbed based on the Syndesi testbed of the TCS-Sensor lab at the University of Geneva, presented in detail in 3.3.2 of the previous chapter. Using Syndesi WSN resources, a dataset consisting of 300'000 measurements was built to be used in the value assignment phase of the virtual resources. In the following sections, we first discuss related work before presenting the mechanisms underlying the modelled testbed framework in detail.

4.2 Related Work

Several testbeds of varying size, hardware, topology and degrees of flexibility have been deployed around the world to facilitate experimental research. Most of the existing testbeds (IoT-Lab [96], WISEBED [97], NetEye [98], SmartSantander [99], FlockLab [100] and more) provide web interfaces for job scheduling, in order to specify the resources needed and coordinate node behaviour as well as data collection. IoT-Lab provides a selection of resources either by Id, or by specific property (location, radio chip, mobility, etc). FlockLab, and IoT-Lab use XML or JSON files to store node configuration and the specifications of an experiment, while other testbeds such as WISEBED, provide their own generic XML-based language, customized so as to improved efficiency.

As previously mentioned, the scope of each testbed can vary, with some facilities focusing on large-scale provision, others on mobility while others target specific environment types and more [101]. For instance, the SWiMNet testbed [102] provides a specialized framework for parallel simulation of wireless and mobile Personal Communication Service (PCS) networks, allowing for a sophisticated modelling of network deployment, traffic and node mobility. In a purely software-defined approach, authors examine the provision of a web-based simulated testbed, focusing on issues such as the interactivity between remote scheduling control systems with a local simulation core. [103]. In another example, Mint-m's experimentation facility MOVIE (Mint-m cOntrol and Visualisation InterfacE) [104] invests in providing the user with a designated interface for interactive control over the testbed as well as a real-time visualization of the testbed activities.

It is apparent that a single testbed cannot fit all user demands, in the various research topics. An approach looking alleviate the limitations of each testbed consists in the federation of individual facilities into unified meta-testbeds. This way, researchers can be granted access to multiple, eventually heterogeneous, testbeds, which allows them to undertake broader and more diversified studies. Notable federation initiatives include among others OneLab [105], IoT Lab [96] and Fed4fire [95]. The OneLab project is one of the first attempts for a flexible federation of testbeds, with their resources being exposed openly to the web. GENI is more focused on networking aspects, services, and security, providing corresponding tools. The Fed4FIRE is a large-scale ongoing project funded by EU which integrates a diverse set of IoT testbeds around Europe (20 at the time of writing), enabling experiments combining their resources.

All of the efforts mentioned above, as well as others in the literature, are aimed at either federating multiple physical testbeds or at providing simulated frameworks as testbeds. This work, on the other hand, focuses on the integration of physical and simulated component into a unified framework, with careful consideration on how to employ virtual resources in a meaningful way.

4.3 Testbed Visualization

Traditionally, sensor testbeds are visually represented with the help of 2D floor plans, where sensor positions are marked on top of a map (an example of a floor map representation in the Syndesi testbed in Figure 4.4). Along the same line, visualization tools employed in network emulator software utilize two-dimensional models to depict the network. We argue that 3D coordinates are necessary when storing sensor locations in testbed representations, since various algorithms may require spatial information (such as euclidean distance) and therefore will perform poorly unless the sensors are co-planar (which is rarely the case in real-world installations). Moreover, provision of the 3D shapes of objects in the testbed space can aid experiments where for instance line of sight is of essence, or when some obstacle-sensitive sensed value distribution in the space is calculated (e.g. illuminance). Finally, by navigating in the 3D model before the running of an

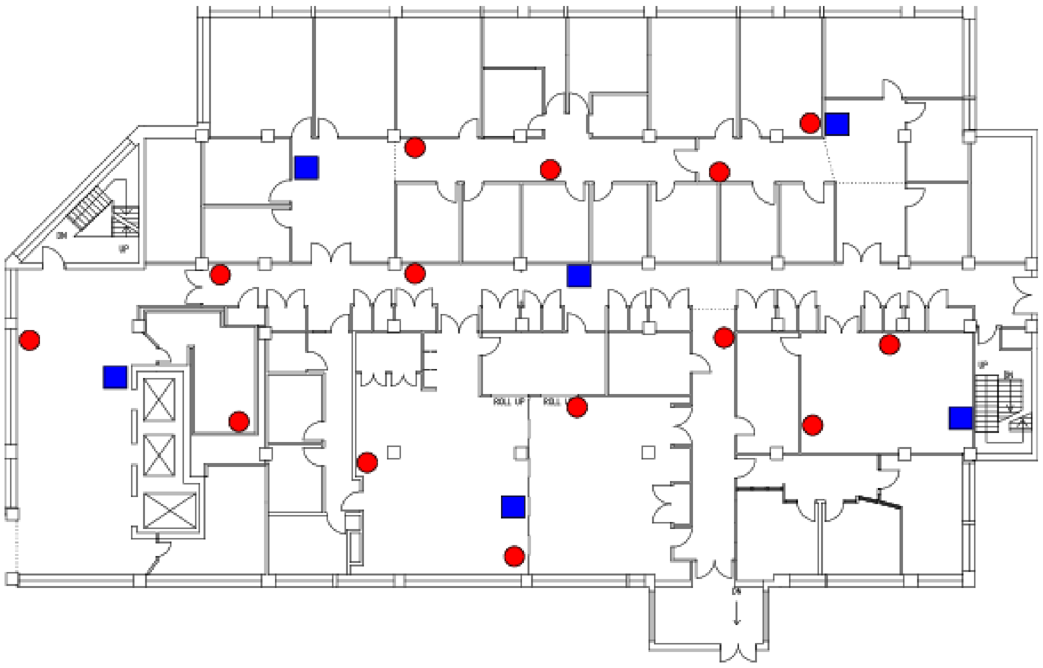


Figure 4.1: A traditional testbed representation on 2D floor plan [2]

experiment, a researcher can capture the intrinsic characteristics of the testbed environment and therefore design the experiment more appropriately.

In order to recreate in 3D the entire infrastructure, including building parts and interior objects, in conjunction with sensor spatial information, we employ an intuitive model developed with the help of the Unity engine. In this model, every component of the surrounding environment, whether a wall, floor, ceiling or a piece of furniture, is stored using six variables: a tuple of spatial coordinates (x,y,z) and a corresponding scaling tuple, expressing the scaling degree on each of the x,y,z axis. Using the above values, we are able to fully reconstruct the testbed environment in 3D, which composes the background tile where testbed sensors/actuators can then be added. For adding resources in this modelled environment, no scaling factor is required but only spatial coordinates together together a single value representing the resource type. For sensors resources, a close-to-live measurement value is also displayed next to their icon in the 3D model, which is fed via the tested management platform APIs. In Fig. 4.3 we see a screenshot of the modelled Syndesi testbed, focused on an illuminance sensor that is placed on a wall next to a window.

The only difference when it comes to physical and virtual sensors in the modelled testbed is the way they are assigned their measurement values in the model. Since virtual sensors do not correspond to a physical entity and therefore cannot get live values from testbed APIs, their measurement values are assigned through a mechanism combining extrapolation of the live physical sensor values and and association analysis based on past data. In the following section we present this mechanism in detail. From an external perspective, since we use the same mechanism for storing information of virtual as well as physical sensors, both components are identical as their existence is blended and their use in algorithms is interchangeable.

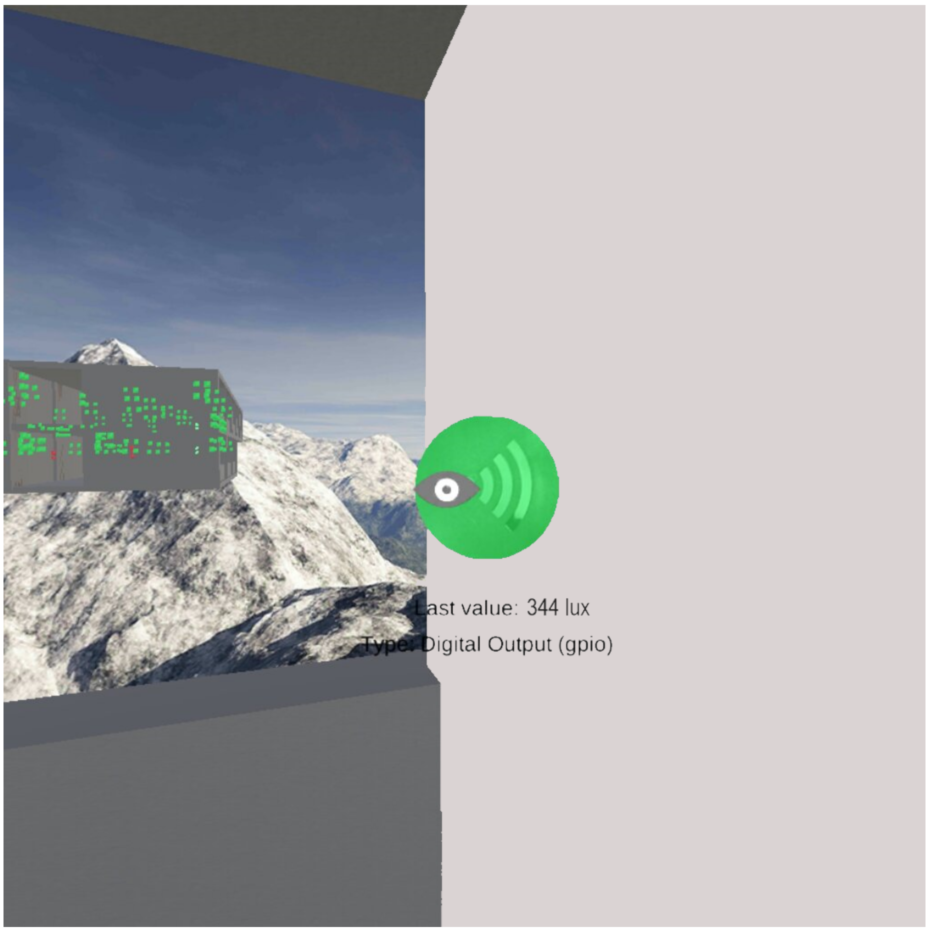


Figure 4.2: Screenshot of the 3D interface modelling the Syndesi testbed. In this instance an illuminance sensor displays its last measured value.

4.4 Augmentation with Virtual Resources

The 3D-visualization interface allows researchers, not only to navigate in the testbed environment, but also to create on the fly virtual sensors to be added in the modelled testbed, in a user-friendly way. In particular, virtual sensors can be spawned with one click when holding down a designated button and pointing to a desired location. Since the aim is to emulate physical sensor functionalities, virtual sensors can be generated only at places physical ones could exist, e.g. on a desk or on the wall, but not for instance mid-air. Upon their formation, virtual sensors can be edited via facilitating features provided by our interface such as drag and drop, sensor type configuration upon right click, sensor deletion and more. Once a modelled testbed is augmented with virtual resources, it can be saved as a separate entity on back-end infrastructure; the saved instance can then be used as a target testbed for an experiment or re-loaded again for further inspection and/or editing. Thanks to this architecture, a single parent physical testbed can be used as a template for the generation of an unlimited number of user-personalized modelled testbeds, which can help researchers meet specific research goals.

In order to ensure a qualitative augmentation of the testbed resources, as much as a quantitative one, the estimated values of virtual sensor measurements must be realistic. To achieve that, we first utilized a simple extrapolation method to calculate those values based on the physical sensors last measurements, but the results were poor. This can be explained due to a non-linear distribution that a sensed parameter can exhibit over the 3D space. Illuminance for instance, depends highly on the space topology, with respect to the position of the light sources such as the sun going through a window, or a lamp that is turned on during office hours. Moreover, a fixed obstacle such as an office separator introduces points of extreme discontinuity of the distribution over space. In order to capture such intrinsic constraints of a given space, we focus on two office spaces belonging to Syndesi testbed, where we collect and analyze an extensive dataset of sensor measurements over a period of approximately 4 months.

In the following sections we detail the process of forming the dataset, before presenting the methodology for assigning measurement values to virtual sensors.

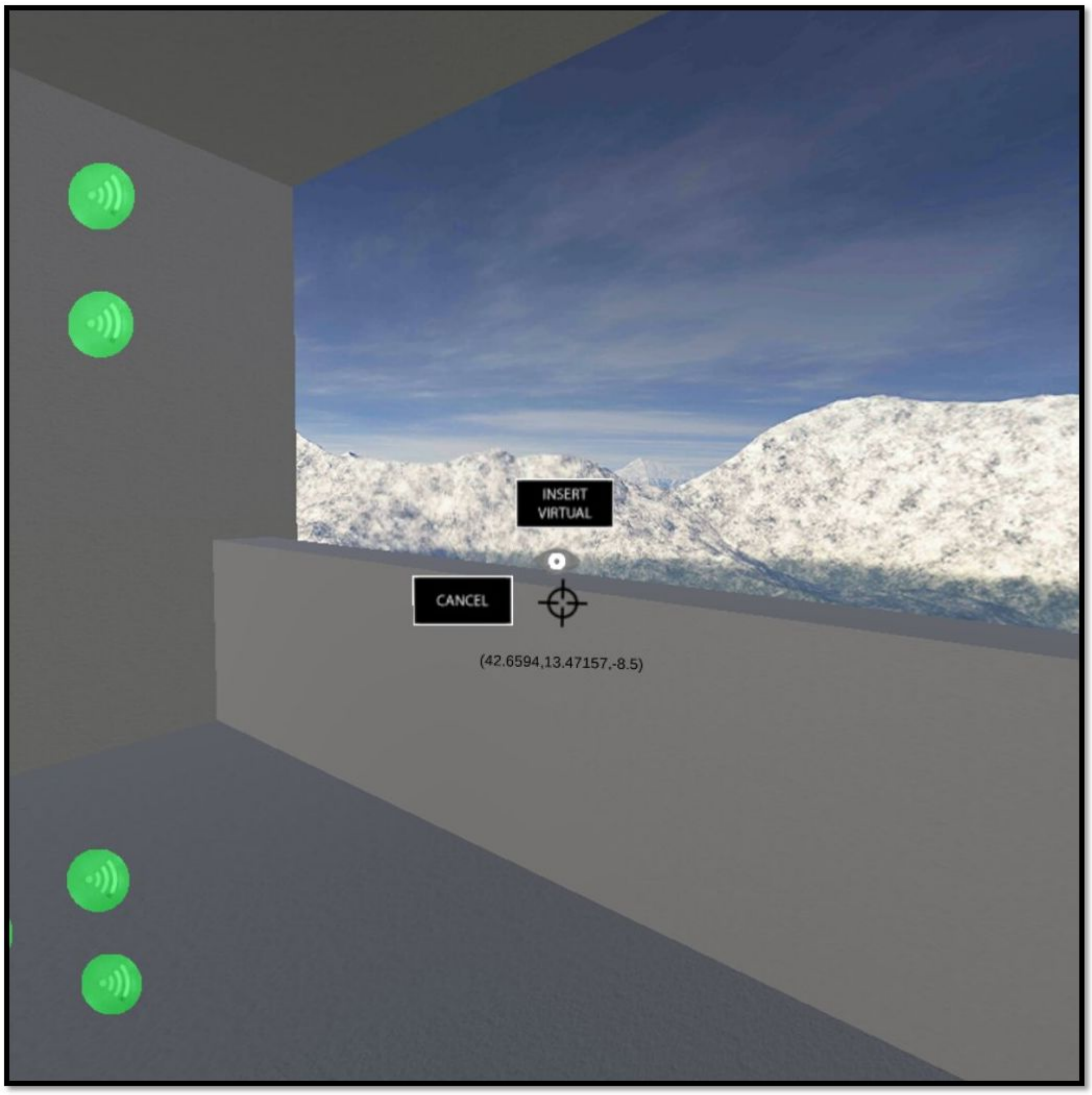


Figure 4.3: Screenshot of 3D interface of the modelled Syndesi testbed. In this instance the user is about to insert a virtual sensor.

4.4.1 Dataset collection

The purpose of forming this dataset was to extract assessments, based on scientific data, regarding environmental parameters and their distribution in indoor space and time. To that end, a designated sensing unit integrated in the Syndesi testbed infrastructure was deployed, composed of 12 sensors divided in two offices of the TCS-lab at the University of Geneva. The rationale of positioning the sensors in space was to be as homogeneously as possible, but some compromise was made in order to ensure securely fixed positions over the period of collection. Under this light, the sensors were positioned along the office walls at a height of 2.4 meters so as interference from daily office routines would be minimal.In the figure below the TCS-lab floor map along with the sensor locations is depicted:

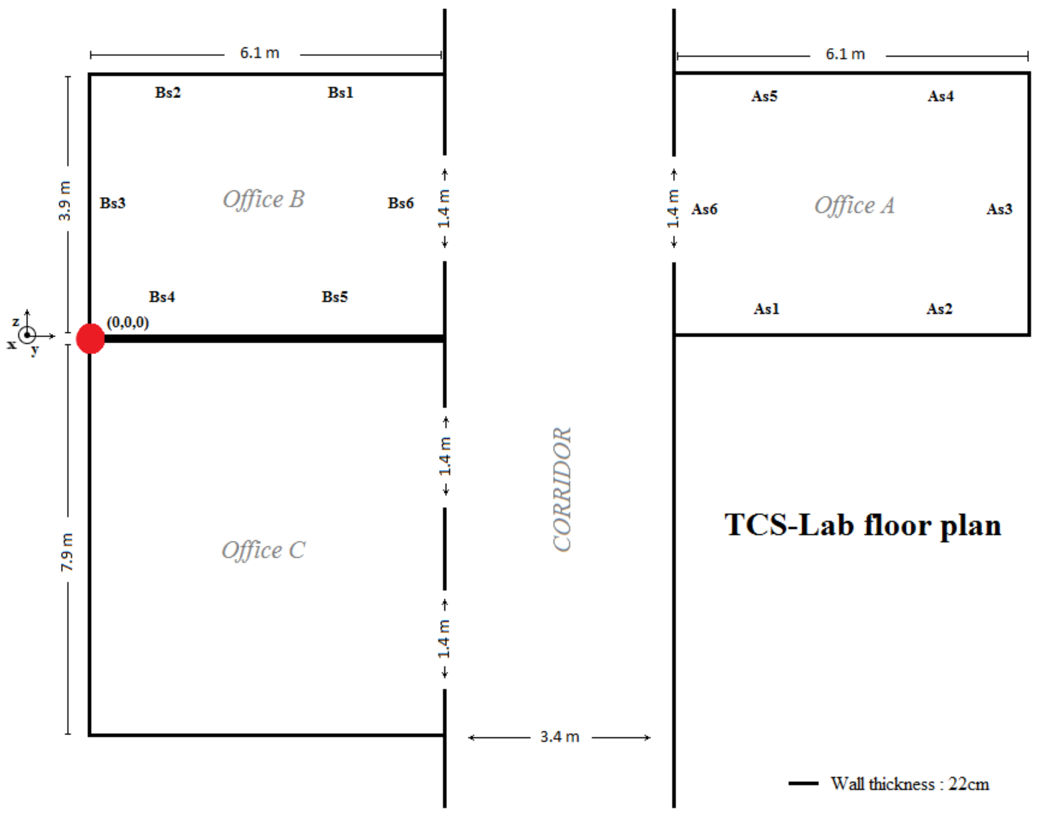


Figure 4.4: A sensing unit of the Syndesi testbed with the purpose of collecting a dataset. Sensors are indicated with their office letter (A,B) followed by their id (s1-6)

The exact positions of the sensors, corresponding to the coordinate system defined in Fig. 4.4 are as follows:

As1 = (2.4, 11.55, 0)	Bs1 = (2.4, 4.05, 3.9)
As2 = (2.4, 13.55, 0)	Bs2 = (2.4, 2.05, 3.9)
As3 = (2.4, 17.6, 1.45)	Bs3 = (2.4, 0, 1.45)
As4 = (2.4, 13.55, 3.9)	Bs4 = (2.4, 2.05, 0)
As5 = (2.4, 11.55, 3.9)	Bs5 = (2.4, 4.05, 0)
As6 = (2.4, 9.05, 1.45)	Bs6 = (2.4, 6.1, 1.45)

Two different types of sensor motes were used in this deployment, the XM1000 from AdvanticSys and the Zolertia Z1 (Figure 4.5). The environmental metrics chosen for the dataset collection are **temperature**, **humidity** and **illuminance**. While the XM1000 is equipped with built-in sensors for all of the above, the Z1 only posseses a built-in sensor for temperature. Nevertheless, a humidity and a temperature sensor can be added via its designated expansion ports (phidgets). According to the manufacturer, external sensors added via the phidget ports necessitate calibration of the readings with appropriate normalization functions, which was effectuated, and the end result was also validated by comparing with the values recorded by the XM1000.

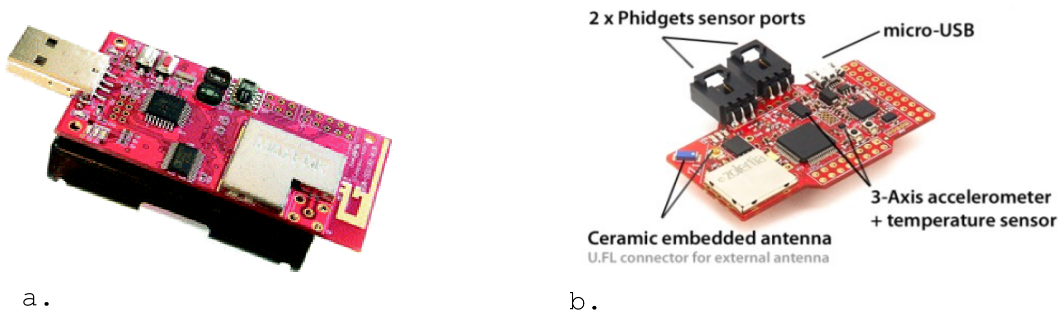

2 x Phidgets sensor ports
micro-USB
3-Axis accelerometer + temperature sensor
Ceramic embedded antenna
U.FL connector for external antenna
a.
b.

Figure 4.5: Sensors used in the sensing unit for the dataset collection: a)AdvanticSys XM1000 [3] and b)Zolertia Z1 [4].

In order to periodically gather the sensor measurements into a database in an automated way, we used a customized back-end script utilizing the Syndesi APIs. Thanks to Syndesi's RESTful architecture each node's resources are exposed as a service allowing for instance sensor measurements to be pulled using a simple URI POST action. The frequency of polling was decided to be 20 minutes, considering the trade-off between time series data granularity and their volume. The data collection started on 15/09/2016 and ended on 31/12/2016, with the final dataset containing of a total of 295.536 measurements, from which 94.140 were for illuminance, 103.062 for temperature and 98.328 for humidity.

4.4.2 Virtual sensor value assignment

To estimate the values of the virtual sensors spawned by users in the modelled testbeds, an incremental approach was followed. First, looking at the problem in a no-memory fashion we calculate their values at a given time t using information only from that time, i.e. the values of all the physical sensors of the testbed at time t.

We denote a physical sensor as:

$$s_i(x_i, y_i, z_i), \quad s_i \in S = \{s_1, s_2, ..., s_n\} \tag{4.1}$$

where x_i, y_i, z_i are the spatial coordinates and S is the set of n physical sensors that belong to the testbed. Similarly, a virtual sensor is defined as:

$$v_i(x_i, y_i, z_i), \quad v_i \in V = \{v_1, v_2, ..., v_m\} \tag{4.2}$$

where V is the set of m virtual sensors created so far in the modelled testbed.

Sensor measurements for temperature, humidity and illuminance are denoted respectively as $s_i(tmp)$, $s_i(hum)$ and $s_i(ill)$, and likewise for virtual sensors. When a new virtual sensor v_k is spawned, for instance in office A and of type $s_i(ill)$, we calculate its measurement value using a weighted average of the physical sensor set a that is located in the same office, such as:

$$v_k(ill) = \frac{w_1 s_1(\dot{i}ll) + w_2 s_2(\dot{i}ll) + ... + w_n s_a(\dot{i}ll)}{\sum_{i=1}^{n} w_i} \tag{4.3}$$

where the weights w examined in the above equation were the inverse and the inverse square of the euclidean distance of v_k with each $s_i \in S$.

This initial approach performed well for temperature and humidity but did quite poorly when it comes to illuminance, indicatively the average Mean Absolute Percentage Error (MAPE) levels were above 20%. This is an expected result, as this approach equally weighs all physical sensors based on the distance, assuming a perfectly homogeneous space distribution of the sensed variable, which in the case of illuminance is rarely the case. Therefore, in order to improve the overall accuracy we decided to make use of the past collected data, with the goal of identifying relational patterns between groups of physical sensors and mapping the space accordingly. In that way, newly spawned virtual sensors, depending on their location, would only be associated with a specific subset of the physical ones, which would lead to better results in their value assignment. In order to extract the correlating physical sensor subsets a brute force approach was followed, using the equation 4.3 and the collected dataset.

The exact methodology is depicted in the following pseudocode:

```
1: for $i = 1, \ldots, k$ do
2:     $S' = S - s_i$
3:     for each time $t$ in dataset do
4:         for each $S'' \in P(S')$ do   ▷ generate powerset and check for all subsets
5:             $E(t, i, S'') = v_i(S'') - s_i$   ▷ calculate error of $v_i$ produced by eq. 4.2
6:         end for
7:     end for
8:     $i, S'' \rightarrow \text{argmin}\{\bar{E}(i, S'')\}$   ▷ keep the subset with the least average error
9: end for
```

Figure 4.6: Pseudocode describing the process of sensor subset association.

Using the above method, we extract clusters of sensors that are more correlated to each other, with which we can segment the space in higher correlated areas. That way, when a new virtual sensor is spawned, depending on its location and the proximity to the physical sensors, it will get its values based on the corresponding subset. Figure 4.7 shows the segmentation of office A, after the association analysis on the past data for the case of illuminance. Since all physical sensors are fixed on the same height (2.4m) the segmentation is two-dimensional, otherwise a 3D represantation would have been necessary.

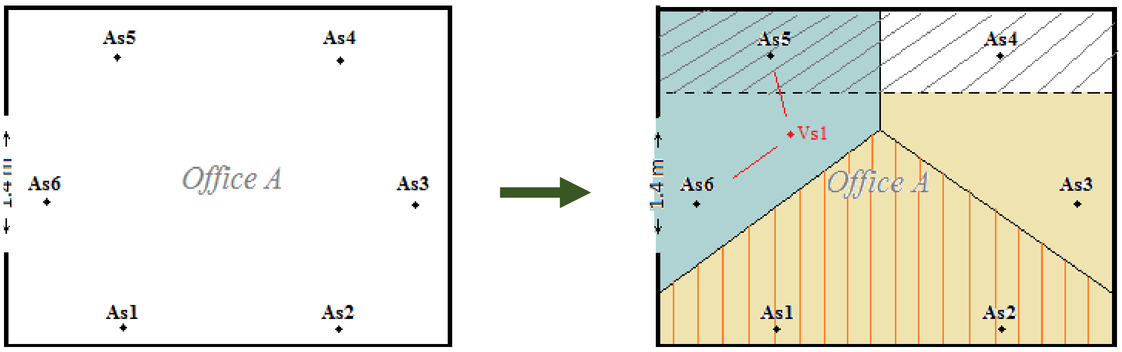


Figure 4.7: Office A map segmented in clustered regions of illuminance after association analysis. On the right, we see a newly spawned virtual sensor (Vs1) getting assigned to the cluster (As5,As6).

We note here that the above methodology does not scale well, considering that for a total of a sensors there are $2^a - 1$ possible subsets. For this study case of 2 offices of 6 sensors the brute force approach is computationally affordable, nevertheless for larger scale infrastructures different, more computationally efficient, methods are envisioned.

4.4.3 Evaluation of Accuracy

In order to evaluate the accuracy of the methods described above, a 10-fold cross-validation was applied on the full dataset. Fig. 4.8 depicts the overall procedure, comprising of a training and a testing phase. First, for each sensor type, the total

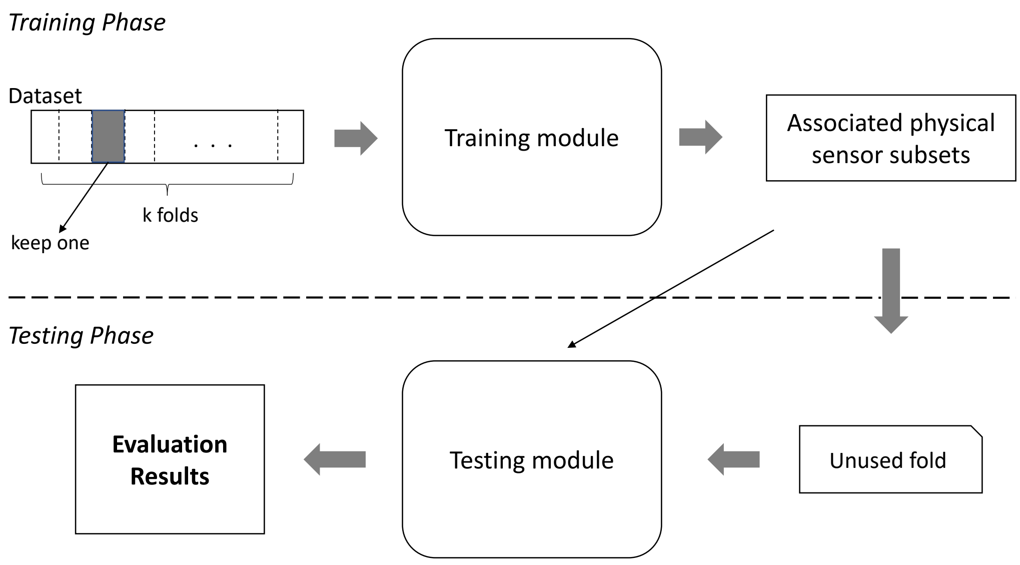


Figure 4.8: Evaluation procedure.

measurements for this type is divided in 10 folds, as equally as possible. Then, during the training phase, one fold is removed and the procedure to extract the correlated subsets (described by the pseudocode in fig. 4.6 is applied to the rest of the folds. In the testing phase, the values of the remaining fold are first computed using the equation 4.2 and the sensor subsets derived in the training phase, and the results are compared with the actual values. This process was repeated until all folds were selected respectively for testing.

The evaluation metrics assessed were the Root Mean Square Error and the Mean Absolute Percentage Error (MAPE). The overall results from the 10-fold cross-validation can be seen in Table 4.1. First, we observe that the difference between inverse distance and inverse distance squared as choice of weights is minimal, with the latter being slightly better. Another observation is that very strong results are acquired with this method when it comes to temperature and humidity, with Mean Absolute Percentage Error (MAPE) values around 1.5%. This is an important assessment, as such low values of MAPE ensure the credibility and scientific correctness of experiments conducted in our modelled testbed. Illuminance, which is a tougher value to predict, produced nevertheless promising scores, with MAPE

Table 4.1: Average Root Mean Square Error and Mean Absolute Percentage Error scores after a 10-fold cross validation.

Weighted average over inverse distance						
Sensor:	Illuminance(lux)		Temperature(^oC)		Humidity(%)	
Room	Average RMSE	Average MAPE	Average RMSE	Average MAPE	Average RMSE	Average MAPE
Office A	9.22 ± 0.55	3.02 ± 0.12%	0.33 ± 0.01	1.07 ± 0.02%	0.60 ± 0.02	1.50 ± 0.01%
Office B	44.4 ± 3.21	6.58 ± 0.20%	0.60 ± 0.01	1.91 ± 0.03%	1.17 ± 0.10	2.61 ± 0.05%
Weighted average over inverse distance squared						
Sensor:	Illuminance(lux)		Temperature(^oC)		Humidity(%)	
Room	Average RMSE	Average MAPE	Average RMSE	Average MAPE	Average RMSE	Average MAPE
Office A	9.39 ± 0.60	3.20 ± 0.14%	0.30 ± 0.01	0.98 ± 0.01%	0.58 ± 0.01	1.42 ± 0.01%
Office B	43.69 ± 3.41	6.57 ± 0.19%	0.61 ± 0.02	1.88 ± 0.04%	1.18 ± 0.10	2.62 ± 0.06%

levels not exceeding 7% , which is still a positive result regarding the quality of the virtual sensors. Finally, the considerable difference in illuminance scores between the two individual rooms (3% MAPE for office A and 7% for office B) can be explained by variations of sunlight exposure due to orientation, which can result in temporal discontinuities that are significantly harder to predict.

4.5 Conclusion

In this work, we introduced the concept of modelled testbeds, which is based on combining physical testbeds with simulation frameworks. Syndesi, a physical WSN testbed at the University of Geneva, was modelled using an interactive GUI that allows users to add virtual resources. The biggest challenge of adding virtual sensors on a physical testbed is to find an adequate mechanism to assigns measurement values as realistically as possible. This is crucial in order to maintain the scientific credibility of experiments conducted in modelled testbeds. To that end, simple extrapolation strategies were initially examined based on the values of the testbed's physical component. This initial approach may perform well for sensor types that have even distribution in space, such as temperature or humidity, but do not have respective results on more unstable variables such as illuminance.

In order to improve the accuracy of the value assigning methods, a dataset of almost 300k measurement was collected, with which relational patterns between physical sensors were identified in a rigorous way. Using that patterns, an improved extrapolation method was employed to assign virtual values based only on a subset of physical sensors, depending on location. To evaluate the accuracy of the methods, a 10-fold cross validation was performed to on the complete dataset. Results regarding sensors measuring ambient temperature, illuminance and humidity, show that extrapolation using a weighted average over inverse distance squared, on the appropriate subset of physical sensors provides optimal accuracy. The average RMSE and MAPE scores using this enhanced extrapolation were sufficiently low to suggest that experimental research using modelled testbeds can be a fruitful field.

Part IV

Crowdsourced Systems

Summary

The final part of this thesis concerns crowdsourcing and crowdsourced systems. Crowdsourcing can be defined as the practice of collecting and aggregating necessary information, services or other types of resources provided by the general public. The systems described in part II, all employed to a certain degree crowdsourcing practices, be it in a sub-system or for a specific functionality. For instance, the mobile application attached to the WSN testbed presented in chapter 3 included a mobile crowdsensing unit, which gathered data from office occupants to the testbed server collected via their smartphone built-in sensors. Similarly, the aggregation mechanisms unifying resources from different testbeds as well as integrating virtual ones described in chapter 4, follow the same principles applied in crowdsourced data aggregation. Consolidating on the above, we proceed to delve deeper to the domain of crowdsourced systems, focusing at the domain of networks and in particular IoT. First, we proceed to identify the formal characteristics, architecture and requirements of crowdsourced systems, exemplifying them in a use case concerning crowdsourced cloud computing. Then, we present a work proposing a novel framework blending crowdsourcing and edge computing, implementing the latter as a community-driven crowdsourced system. Finally, using the insights gained throughout the rest of the thesis, we build a networking and computing framework for crowd collaboration using smartphones in local proximity.

5 Definitions, Architecture and Requirements of IoT-related Crowdsourced Systems

5.1 Introduction

The practice of interacting with the crowd for reaching a common goal in cases such as problem solving, process optimization, product innovation etc. is becoming increasingly popular in the modern world. Crowdsourcing is employed across many different industries and areas, such as design, IT, health and science, finance, marketing and many more. The term crowdsourcing is used quite loosely in the literature, nevertheless a converging definition is that it is the process of obtaining services, ideas, or other content by soliciting contributions from a large group of people, rather than from traditional employees or other suppliers. In one of the prevalent interpretations, crowdsourcing refers to the distribution of tasks that require some human intelligence, also mentioned as Human-Intelligence-Tasks (HIT), to the crowd. Thanks to the emergence of online crowdsourcing platforms, such as Amazon Mechanical Turk [1] or CrowdFlower [2], which provide the means for reaching the crowd and are application-agnostic, this practice has seen significant growth in the recent years.

The benefits from the practice of crowdsourcing are many and of diverse nature. First, there can be significant savings in budget compared to the approach of

[1]https://www.mturk.com/

[2]https://visit.crowdflower.com/People-Powered-Data-Enrichment_T

strictly involving professionals for a desired goal. Moreover, there is a qualitative aspect brought in by engaging the crowd; this is intuitively witnessed for instance in the popular game “Who Wants to Be a Millionaire?” where players are given the option to ask the crowd to answer a question for them and, while any random vote does not mean much, taken together the collective verdict tends to be very accurate. This added value, although, is bound to certain requirements that the crowd must fulfill; James Surowiecki in his book "Wisdom of Crowds" identifies four distinct criteria that must be met for a crowd to be "wise", namely diversity, independence, decentralization and aggregation [106].

In the field of IoT, the proliferation of portable smart devices that are equipped with built-in sensors, such as smartphones and smart watches, has enabled a particular crowdsourcing paradigm, referred to as mobile crowdsensing. In mobile crowdsensing, sensed environmental data from large areas are gathered via the crowd’s devices, complementing data collected from traditional fixed IoT sensors. Moreover, the emergence of affordable Single-Board-Computers (SBCs) for personal use, such as Arduino and Raspberry Pi, provides another mean for crowdsourcing IoT data. In order to facilitate such systems, specialized crowdsourcing platforms have been implemented for mobile crowdsensing (e.g. gMission [107]), providing supporting mechanisms and/or tailored environments for application development. A distinction needs to be made from such crowdsourcing platforms and *crowdsourced* systems. Crowdsourced systems, contrary to the enabling platforms, are systems whose constituent infrastructure is pooled or augmented via crowdsourcing methods. Therefore, crowdsourced systems may or may not include the use of a crowdsourcing platform in their architecture; we can picture them as a higher level entity in a corresponding ontology.

Besides mobile crowdsensing systems (MCS), another field of prevalence of crowdsourced systems is collaborative computing, also referred to as volunteer computing [108]. Volunteer computing enables individual users to share their computer’s idle processing power, forming that way an inexpensive high-performance parallel computing cluster. In the field of emergency management, crowdsourced systems enbaling the formation of ad-hoc networks using crowd devices in order to substitute damaged infrastructure have been proposed [109]. It is certain that several technical challenges are imposed when utilizing crowdsourced resources; for instance, the contributed resources are typically characterized by a high degree of heterogeneity due to the opportunistic manner that these are provided by the crowd. For the same reason, crowdsourced resources also demonstrate a significant variance in their availability and reliability rendering the robust design and provisioning of a desired service a challenging task. The challenges in crowdsourced systems include as well the design of adequate incentive mechanisms for the crowd, in order to ensure sufficient participation. There has been significant research focusing on this issue, where different types of incentives are identified: intrinsic motivators, such as having fun or taking on a challenge, as well as extrinsic motivators such as financial reward, fame or social pressure.

In this work, we examine IoT-related crowdsourced systems in a formal manner; we define their functional architecture, identifying its underlying components, and we outline the corresponding requirements. Before presenting the study results regarding the "canonical" crowdsourced system, we provide some necessary background and discuss the state of the art in the following section.

5.2 Background

5.2.1 The Practice of Crowdsourcing

The first references of crowdsourcing practices point back to the 18th century, where the head of states in Britain and France disseminated open calls to the public, in order to gather solutions to problems that individual experts could not adequately solve [110]. In these early examples, the crowd's added value was leveraged in a form of a competition where only the best solution is picked, hence no aggregation of individual contributions took place. In another early crowdsourcing example, in 1906, a researcher at a country fair gathered 787 guesses concerning the weight of an ox and, astonishingly enough, the average value of the crowd's guesses was one pound short of the actual weight [111]. In this case, we notice that it was the combination of all individual guesses, which were of no worth if singled out, that derived the value of the crowdsourced solution. This brings us to the problem of classifying the different forms of crowdsourcing, which does not have an easy solution, since crowdsourcing covers a relatively diverse set of practices. One of the reason it is difficult to provide a cover-all categorization (seen also by the diverging results in the literature) lies on the choice of classification criteria, that can greatly vary (e.g. types of tasks, nature of incentives etc). Nevertheless, a relative convergence can be found in a slightly broad categorization which identifies four distinct forms of crowdsourcing: 1) crowd labour 2) crowd creation 3) crowd voting and 4) crowd funding.

Crowd labour refers mostly to microtasking, which is a form of crowdsourcing in which work is decomposed into a set of small, self-contained tasks, each typically able to be completed in a matter of minutes. The approach has been used to address a number of different problems, ranging from labeling images to handwriting transcription. In the case of crowd creation, the task at hand is of a more creative nature, be it a design approach or a challenging scientific problem, and the crowd is asked to contribute with ideas or attempts towards a solution. Among popular examples of crowd creation is Threadless [3], where consumers propose new T-Shirt design ideas, or iStockPhoto [4] in which amateur photographers contribute high quality stock photography images, while open source software development also falls into this category. Crowd voting uses the collective wisdom of the community to evaluate content such as articles, music media and more. It is one of the most popular forms of crowdsourcing, having the highest levels of engagement. Finally, crowd funding enable the funding of a project by raising small amounts of money from a large number of people, typically via an open call on a platform on the Internet, circumventing that way the traditional corporate establishment and

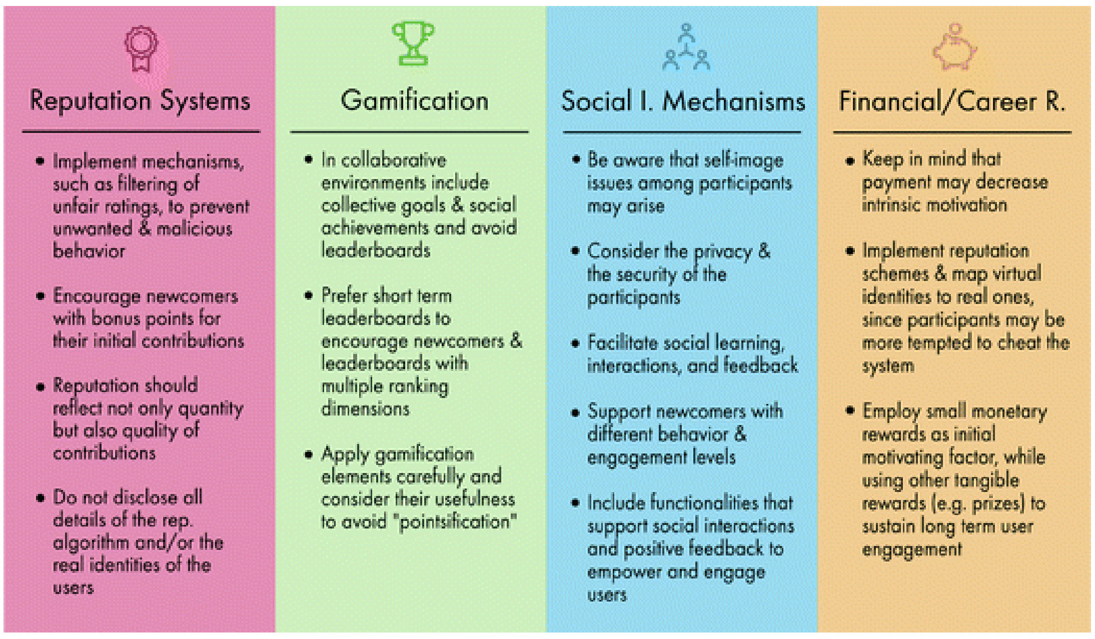
[3]https://www.threadless.com/

[4]https://www.istockphoto.com/

offering financing to individuals or groups that might otherwise be denied credit or opportunity. Kickstarter[5] and Indiegogo [6] are prominent examples of crowdfunding platforms, with many more also being widely used.

In all forms of crowdsourcing, it is paramount that some incentive is provided to the crowd in order to ensure its participation in a consistent way. Incentives correspond to the added value perceived by crowd participants for supplying their contribution to the system. They can be categorized in two broad types, namely intrinsic and extrinsic incentives. Intrinsic incentives reward participants via things such contentment, for instance when contributing to a greater cause for the common good, or self achievement, skill development and likewise. Extrinsic incentives consist in offering more direct and tangible rewards, such as monetary returns or offered services. Various mechanisms have been proposed to fullfil the incentivisation process; Katmada et al provide a simple yet accurate categorization [5], detailed in the following figure.

[5]https://www.kickstarter.com/
[6]https://www.indiegogo.com/



Reputation Systems	Gamification	Social I. Mechanisms	Financial/Career R.
• Implement mechanisms, such as filtering of unfair ratings, to prevent unwanted & malicious behavior	• In collaborative environments include collective goals & social achievements and avoid leaderboards	• Be aware that self-image issues among participants may arise	• Keep in mind that payment may decrease intrinsic motivation
• Encourage newcomers with bonus points for their initial contributions	• Prefer short term leaderboards to encourage newcomers & leaderboards with multiple ranking dimensions	• Consider the privacy & the security of the participants	• Implement reputation schemes & map virtual identities to real ones, since participants may be more tempted to cheat the system
• Reputation should reflect not only quantity but also quality of contributions	• Apply gamification elements carefully and consider their usefulness to avoid "pointsification"	• Facilitate social learning, interactions, and feedback	• Employ small monetary rewards as initial motivating factor, while using other tangible rewards (e.g. prizes) to sustain long term user engagement
• Do not disclose all details of the rep. algorithm and/or the real identities of the users		• Support newcomers with different behavior & engagement levels	
		• Include functionalities that support social interactions and positive feedback to empower and engage users	

Figure 5.1: Categories of incentive mechanisms for crowdsourcing [5]

5.2.2 The Wisdom of the Crowd

In most forms of crowdsourcing, the effectiveness of the collective solution highly depends on the characteristics of the crowd. For instance, in a highly homogenous crowd, cognitive biases can undermine the predictive power of crowdsourcing. Despite the self-evident requirement of magnitude (since a few individuals do not constitute a crowd) there are other key crowd attributes that can predict the quality of the crowdsourced solutions, a subject of study which traditionally falls within the social sciences. Scott Page, in his book "The Difference: How the Power of Diversity Creates Better Groups, Firm, Schools, and Societies" [112] argues that the effectiveness of crowds depends on diversity, not in terms of what we appear from the outside but what we are really like from within, i.e. our individual abilities and tools. Progress and innovation, according to Page, may depend less on high IQ lone thinkers than on a diverse group of people working together and capitalizing on their individuality.

Along the same line, James Surowiecki, in his book "Wisdom of Crowds", points out that "collective decisions are most likely to be good ones when they're made by people with diverse opinions reaching independent conclusions, relying primarily on their private information" [106]. Based on this reasoning, Surowiecki proceeds to identify in his book the four elements that characterize a "wise" crowd:

1. **Diversity**. A diverse crowd is composed by individuals possessing a varying background and characteristics, via which they provide their own contribution/opinion, even if it is an eccentric one.

2. **Independence**. Each crowd member should operate independently and its opinion should not be influenced by anyone else.

3. **Decentralization**. Decentralization refers to individuals being able to specialize and draw on their local knowledge, rather than on an omniscient or farseeing planner.

4. **Aggregation**. The crowd's individual contribution should be aggregated via appropriate mechanisms into one collective decision.

The above delimitation provides a groundwork for determining the crowd requirements that govern crowdsourced systems also in the field of IoT. In a following section, we utilize Surowiecki's work to reach conclusions relevant in the context of IoT network.

5.2.3 Crowdsourcing in IoT

Since the last two decades, crowdsourcing has been applied in the field of IoT in a variety of fields and applications. For instance, in WSN testbeds, as seen in the previous chapter, mobile resources from crowd devices can be integrated in order to augment the testbed's data collection capacity among others. This paradigm, where sensing data is gathered via the crowd, is referred to as mobile crowdsensing and has seen a rapid growth in recent years. In a recent mobile crowdsensing example, FLOAT [113] provides the citizens of Beijing with kites that have specialized sensors kits attached on them, in order to gather environmental data regarding air quality. In the case of FLOAT the design of the customized kite happened as well in a crowdsourced way, in the form of crowd creation. Another crowdsensing project was initiated in Fukushima, Japan, following the nuclear plant meltdown in 2011. There, by using open source and open data approaches, the Safecast project enabled citizens to create their own DIY radiation monitoring sensor kits. That way, the population was given the ability to monitor, gather, and openly exchange information about environmental radiation and other pollutants, which was then collected and visualized in radiation maps.

Apart from collecting sensing data, crowdsourcing is also employed to enable collaborative computing. SETI@home was a 20 year-long project analyzing radio signals to look for signs of extraterrestrial intelligence [114]. In order to speed the heavy signal processing necessary it opportunistically employed volunteer computers processing power, during their idle periods. In a current example, DreamLab crowdsources computational resources from smartphones while they're charging in order to speed up calculations of a machine learning algorithms [115]. In a COVID-19 related DreamLab use case at the Imperial College of London (ICL), it is indicated that, with a nightly runtime of six hours, the performance of 100,000 smartphones would reach the annual output of all ICL computers in just three months [116].

5.3 IoT-related Crowdsourced Systems

A crowdsourced system is a system that employs crowdsourcing in order to augment its constituent infrastructure, offered services or collected information. In the field of IoT, crowdsourced systems have emerged thanks to technological enablers which help the gathering of crowdsourced infrastructure, such as advances in ubiquitous computing and mobile networking. In particular, the wide adoption of smartphones from the general public, as well as the appearance of affordable SBCs, provided a fertile ground for the rise of IoT-related crowdsourced systems. From now on when we refer to crowdsourced systems we address them in the context of IoT, i.e. IoT-related crowdsourced systems.

5.3.1 Architecture

While crowdsourced systems may be built with the intent of accomplishing a specific task (such as FLOAT and Safecast), in the general case crowdsourced systems should be application agnostic. In figure 5.2 we present the high level architecture of a canonical crowdsourced system. Since this architecture corresponds to the general case and is not application specific, it does not delve into technological or implementation details but rather focuses on the fundamental constituent entities and their relationship.

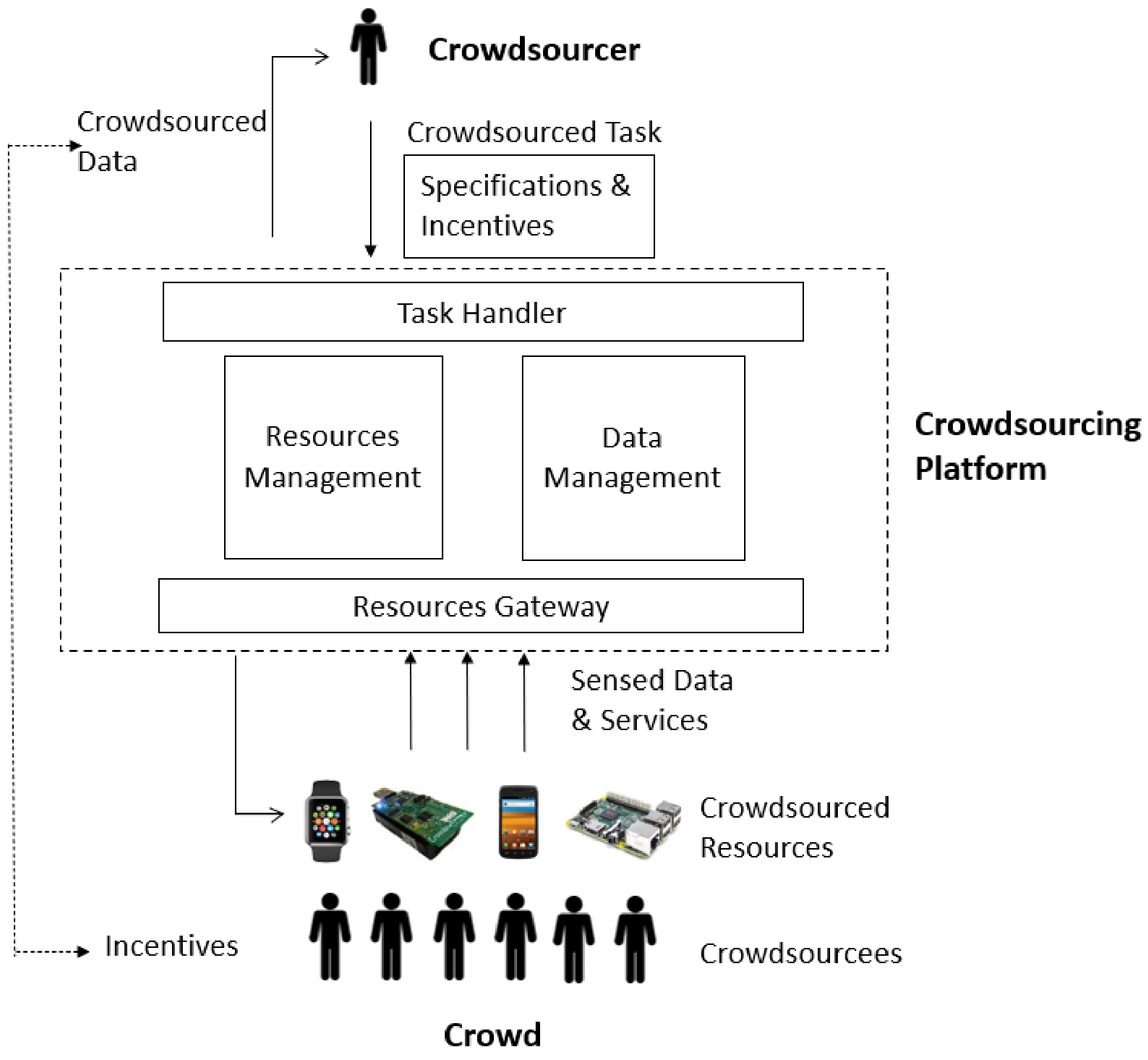


Figure 5.2: High-level architecture of IoT-related crowdsourced systems.

We identify three distinct layers which compose crowdsourced systems: the crowdsourcer, the crowdsourcing platform and the crowd. The crowdsourcer is where the crowdsourced task originates and is also responsible for providing sufficient incentives. In order to reach the crowd, the task goes through a compound entity, the crowdsourcing platform, consisting of separate modules for resources and data management, as well as a task handler and a resources gateway acting as input/output units. We note here that the ensemble of actions performed in this layer is often enclosed and provided by an external entity (f.i. a crowdsourcing platform such as Amazon Mechanical Turk or Kickstarter), but that does not mean that it has to necessarily be implemented that way. As long as the required middleware modules are present to link the crowdsourcer and the crowd, a crowdsourced system can be set up. Once the task has reached the crowd, there is the corresponding trajectory of the crowdsourced data and services from the crowd back to the crowdsourcer, again via the crowdsourcing platform. In the following we detail each of the architectural elements:

- **The crowdsourcer**. The crowdsourcer defines the task to be crowdsourced, prescribing its specifications as well as the corresponding incentive mechanisms. The task specifications may include requirements regarding the crowd resources to be employed, such as resource type or geographic location, depending on the application. The incentive mechanism to engage crowd participation is also defined by the crowdsourcer, which can be of extrinsic nature (e.g. a monetary or service reward) or intrinsic (e.g. feeling of contentment for contributing to a noble cause).
- **The task handler**. The task handler acts as the interface of the crowdsourcing platform to the crowdsourcer. It enables him/her to discover, manage and inspect the crowdsourced resources in a uniform manner via its corresponding functions. It is there where the task specifications and incentive mechanisms are input by the crowdsourcer and then relayed to the resource and data management modules in a proper way for the task to be compiled and executed. Moreover, once results from the crowd have been retrieved it is responsible for returning them to the crowdsourcer in a meaningful format, f.i. results in a semantic context instead of raw values. In general, the task handler can be seen as an abstraction layer that hides the underlying complexity of the crowdsourced system to the crowdsourcer, while facilitating task composition and execution.
- **Resources management**. The resources management module is where all the information and meta-data regarding the system resources is maintained and managed. It operates as a resource directory, supporting all of the services

required for resource discovery, registration and access, exposing them all in a user-friendly way. In addition, it maintains all the meta-information which characterize each resource, f.i its type, method of access, availability, trustworthiness and more. Generally, the resource manager should store all information in a standardized format with the aim of (i) mitigating heterogeneity issues and (ii) facilitating the use of this information by other system elements.

- **Data Management**. Data Management is the module where all acquired data, as well as pertinent meta-data, are kept and curated. For example, if temperature measurements are collected as part of a crowdsensing job, the term "data" would refer to the actual temperature readings, while "meta-data" relates to the information regarding e.g. the time and location where the readings were taken. Similarly to resources management, data and meta-data should be maintained in a standardized format in order to address heterogeneity and interoperability issues. In this regard, technologies used in big data and cloud storage are envisioned in the data management module, to address the case of a crowdsourced system that generates big volumes of data. Finally, depending on the specific application of the crowdsourced system, the services for storing, processing and curating data may be highly sophisticated, with pattern recognition and other data mining mechanisms being often used to extract semantically meaningful information from initial data (e.g. to predict a weather alert from a set of environmental measurements).

- **Resources Gateway**. The resources gateway module interconnects the individual crowdsourced resources with the crowdsourcing platform. This link happens such as the underlying heterogeneity and diversity of the crowdsourced resources are hidden from the upper levels of the architecture. This mitigation is usually achieved with mechanisms implemented in commonly understood interfaces, such as mobile applications for smartphones or software add-ons for SBCs. It is common for a crowdsourced system to provide multiple options to the crowd, all being a part of the resources gateway, in order to facilitate and stimulate participation.

- **The Crowd**. The crowd is the foundation of a crowdsourced system and an indispensable part of its functional architecture. The term can refer either to the crowdsourced resources, i.e. the devices that contribute to the system or directly to the crowdsourcees ,i.e. the owners of those devices. It is worth noting that there is no 1-to-1 relationship between crowdsourcees and crowdsourced resources, as one person may possess more than one device. In the context of a crowdsourced task, the crowd provides sensed data and/or services and in return receives some reward, which may come at various forms (extrinsic or intrinsic incentives).

5.3.2 Requirements

As a starting point in the inspection of functional requirements of crowdsourced systems, we revisit the prerequisites of the "wise" crowd presented in section 5.2.2. The 4 requirements identified by Surowiecki are addressed to a crowd of people, rather than crowdsourced resources; that being said, some parallel can be drawn in the context of IoT-related crowdsourced systems. In such an effort, we translate these 4 requirements as follows:

- **Diversity:** A canonical crowdsourced system needs to be able to incorporate diverse crowdsourced infrastructure. This refers to to possessing adequate integration mechanisms to be able to crowdsource highly heterogeneous devices, without relying on specialized hardware that is application specific.
- **Independence:** All crowd devices should operate in an autonomous way, independent of other system components. There should be no dependencies or the presence of a hierarchical structure among the crowdsourced resources.
- **Decentralization:** There should be no central mechanism dictating the operation of crowdsourced devices. Nonetheless, orchestrating mechanisms which are centralized may exist in order to supervise task execution and manage the crowd resources as a whole.

- **Aggregation:** Aggregation mechanisms should exist which accurately consolidate individual contributions towards achieving some specific goal. Such a mechanism may be executed centrally in a crowdsourcing platform, or it could also be intrinsic to the crowdsourced system, for instance in the form of a distributed protocol.

The above requirements outline accurately important aspects that a crowd of devices, in an IoT-related crowdsourced system, should fulfill. Going a step further, in the following we identify the functional requirements of the whole crowdsourced system, which some overlap with the above conclusions drawn from the prerequisites of a "wise" crowd.

1. **Openness** It is required that an IoT-related crowdsourced system maintains its openness, as the driving force behind crowdsourced systems in general is the solicitation of contributions made by a large, open and diverse crowd. The requirement of openness applies to several aspects of crowdsourced systems, the critical one being a) access to the system, b) design principles and c) data models. In the following we detail these principal sub-categories regarding the openness requirement:

   (a) *Openness in terms of crowd accessibility.* A crowdsourced system must have an architecture built to incorporate publicly accessible interfaces, which allow crowd individuals to participate in large numbers. Websites or smartphone apps, for example, may be provided by the infrastructure to facilitate crowd engagement. While some access limits may be imposed to the crowd depending on the application context (e.g. targeted demographic criteria, inhabitants of a specific area or registered clients of a specific service), they should not limit participation to a very narrow or extremely specific group of individuals, and should be inherent to the task at hand.

   (b) *Openness in terms of hardware and software design.* It is required that a crowdsourced system is created following design principles, in terms of hardware as well as software, that facilitate crowd participation. This is achieved by the use of modular architectures, which allow for future system additions, as well as the utilization of standardized technologies that are widespread in the general population. It is recommended that different development platforms are supported in scenarios where the crowd contributes to the system by developing a piece of hardware (e.g. a sensing device). When crowdsourcees contribute to the system by developing or deploying software, it is advised that the software's hardware requirements are as generic as possible.

(c) *Openness in terms of Data.* A open and standardized data model is necessary in the data management module of a crowdsourced system. This is because, depending on the application, crowdsourced systems may collect data from a wide range of devices with varying calibrations, as it is the case for instance in system monitoring ambient environmental conditions. Similarly, if devices in a crowdsourced system are required to exchange data (e.g., to perform some local data processing), a standardized format is needed to ensure interoperability.

2. **Affordability and availability of system components.** Affordability and availability imply that a crowdsourced system should not rely on specialized equipment. On the contrary, it is required that the necessary system components (hardware or software) are both inexpensive and readily available to the general population. That is because ordinary people should be able to participate using either off-the-shelf equipment, such as smartphones, or piecing together some off-the-shelf components, as it is the case with SBC development platforms.

3. **Anonymity of Crowdsourcees.** A crowdsourced system should employ mechanisms that can protect the identity of the individual crowdsourcees. This refers to the system providing adequate assurances that an individual cannot be identified, via its contributions to the system. This is important as it helps in the mitigation of any concerns of trust and privacy that could hinder participation in the general public. In addition, a crowdsourced system

should not be dependent on the identities of specific individuals by definition, as it is the collective result (which "belongs" to the crowd as a whole) that is of value, and hence by design. While anonymity should be guaranteed, in specific applications a task may nevertheless include filter methods for identifying sub-groups of crowdsourcees based on specific characteristics (e.g. demographics).

4. **Aggregation Mechanisms.** A crowdsourced system must incorporate adequate aggregation mechanism for processing the crowdsourced data. Aggregation methods should treat data independently of the identities of the crowdsourcees, however, they may take into account factors like reputation, trustworthiness, and so forth. Aggregation mechanisms in IoT-related crowdsourced systems belong in two types: a) centralized aggregation mechanisms, in which the data is processed after it has been collected in a data repository, or b) in-network aggregation mechanisms, in which the data is processed locally by the nodes of the crowdsourced system before being sent to the system's data repository. Hybrid solutions, which combine the two categories, may also be implemented.

5. **Abstraction Mechanisms.** A crowdsourced system should have abstraction mechanisms in place for each component of its architecture. These mechanisms serve the purpose of facilitating the integration between all components, by masking their underlying heterogeneity. They also help to clearly distinguish the system participant roles, as in for example, the crowdsourcer, which interacts with the crowd as a whole via designated APIs, rather than directly with individual crowdsourcees. The use of abstraction mechanisms in system development implies that the overall system has a modular design. This contributes towards more efficient system designs overall, which complement as well other requirements (such as the anonymity of the crowd).

6. **Incentive Mechanisms.** An incentive mechanism should be employed by the crowdsourcer in order to successfully engage the crowd in a consistent and meaningful way. The reasons for convincing the crowd to participate in a crowdsourced system stem from a wide spectrum of motives, such as

monetary reward, altruism, social motivations etc. It is the crowdsourcer's responsibility to pick an appropriate incentive mechanism, given the specific application and the targeted crowd.

5.3.3 Crowdcloud: A study case

Following the requirements and architecture outlined and detailed in the previous section, we present here a demonstrative study case of an IoT-related crowdsourced system: CrowdCloud. The CrowdCloud consists of a system providing crowdsourced cloud infrastructure, by the crowd and for the crowd. Therefore, in this case the crowdsourcer and the crowdsourcee are interchangeable, both belonging to the crowd, as we can see also in the CrowdCloud architecture depicted in figure 5.3.

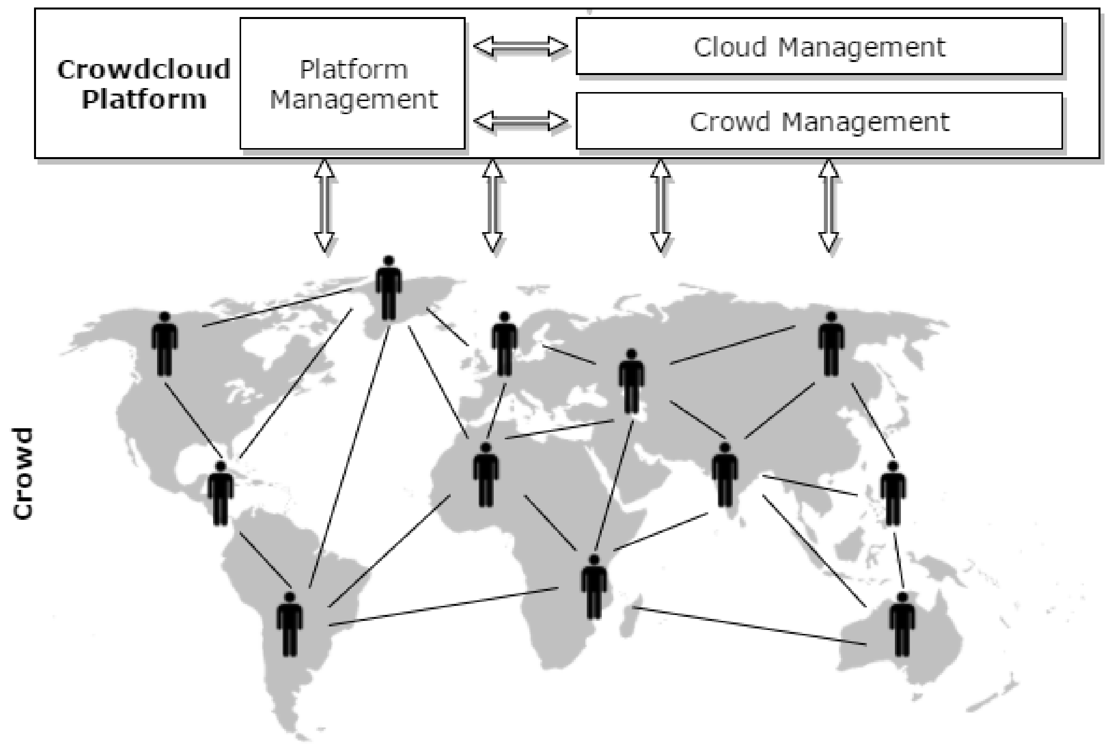


Figure 5.3: CrowdCloud system architecture

Looking at the CrowdCloud platform in figure 5.2, with the canonical architecture of an IoT-related crowdsourced system as a reference (fig. 5.2), the crowd management module refers to the resource management functionalities, such as registering crowd users and keeping track of their status. Similarly, the cloud management module encircles the actions corresponding to the data management component, such as storing user cloud data or manage cloud computations. Finally, the CrowdCloud platform management module abstracts the task handler and resources gateway functionalities, providing the interface for crowd users having either the role of a crowdsourcer or a crowdsourcee.

The idea underpinning the driving force for a crowd-driven cloud revolves around a free market model, where every individual can supply its resources for a price, or equivalently demand for resources from the others, following the regulations that dictate a free market. This encompass the necessary incentivization mechanism, with crowd participants being rewarded with some monetary or service-based reward (extrinsic incentive) for sharing their device resources. The CrowdCloud resources may correspond to the infrastructure level, such as CPU power and storage space, the platform level, e.g. specialized libraries and web servers, or also to the software level such as on-demand software systems. At this stage, we consider the provision of resources at the infrastructure level; to that end, appropriate mechanisms allowing for resource abstraction, task partitioning and parallel execution are necessary to leverage the crowd resources. For the prototype use-case, we choose to use Apache Spark [117] which is a cluster computing software encapsulating all of the above functionalities, providing a distributed application execution environment that is compatible also with resource-constrained devices. This is important in order to address the openness requirement in crowdsourced systems, allowing as many crowd members as possible to participate in the system.

Spark utilizes the MapReduce programming model to partition tasks and is considered pioneer in using in-memory data abstractions that help reach very high levels of speed and scalability [118]. The architecture of a Spark cluster is depicted in figure 5.4. Spark applications operate as separate groups of processes, coordinated by the SparkContext object on the driver program, in a master/slave arrangement. SparkContext can communicate with a variety of cluster managers

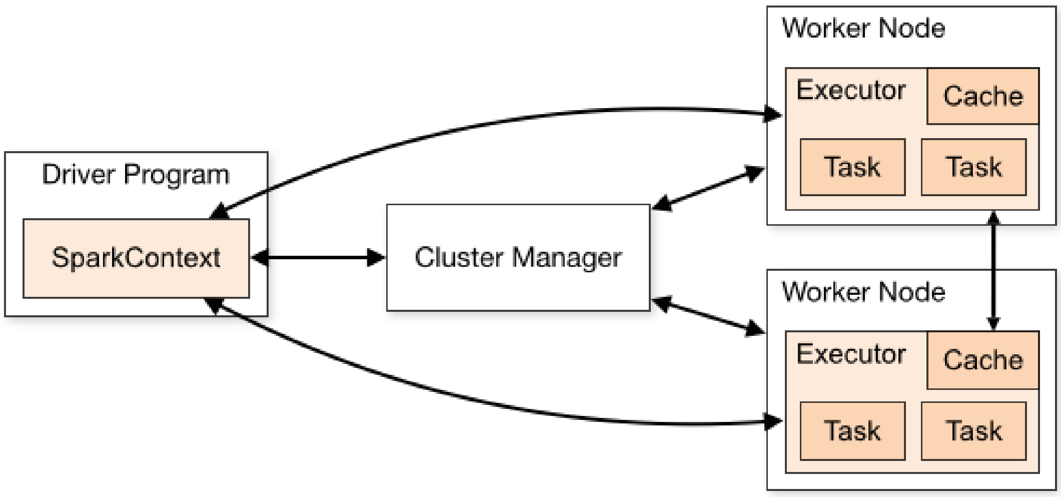


Figure 5.4: Spark cluster architecture

that distribute resources among several applications. The cluster manager connects to the nodes in the cluster and obtains their executors, which are processes that execute computations and store data for applications.

The cloud management module of the CrowdCloud architecture is implemented by the native Spark standalone cluster manager which we use in this prototype. Other open source cluster managers (Mesos, Yarn) are also compatible for use with Spark, but the standalone option is the simplest and most lightweight solution. That way, in order for crowd users to expose their resources in the system, they only need to install the spark-worker image on their device and register with the CrowdCloud platform. Upon successful registration, the cloud manager can then retrieve part of the device's computational cores in accordance with the plan and policy in place. The cluster manager will be responsible for communicating the necessary metadata concerning crowd users, such as the number of available cores, memory etc., to the crowd management module where all the free market functionalities will be contained.

In order to test the validity of the above framework, we deployed it in a local cluster of Raspberry Pi devices (Fig. 5.5). The Raspberry Pi 3 Model B version used in

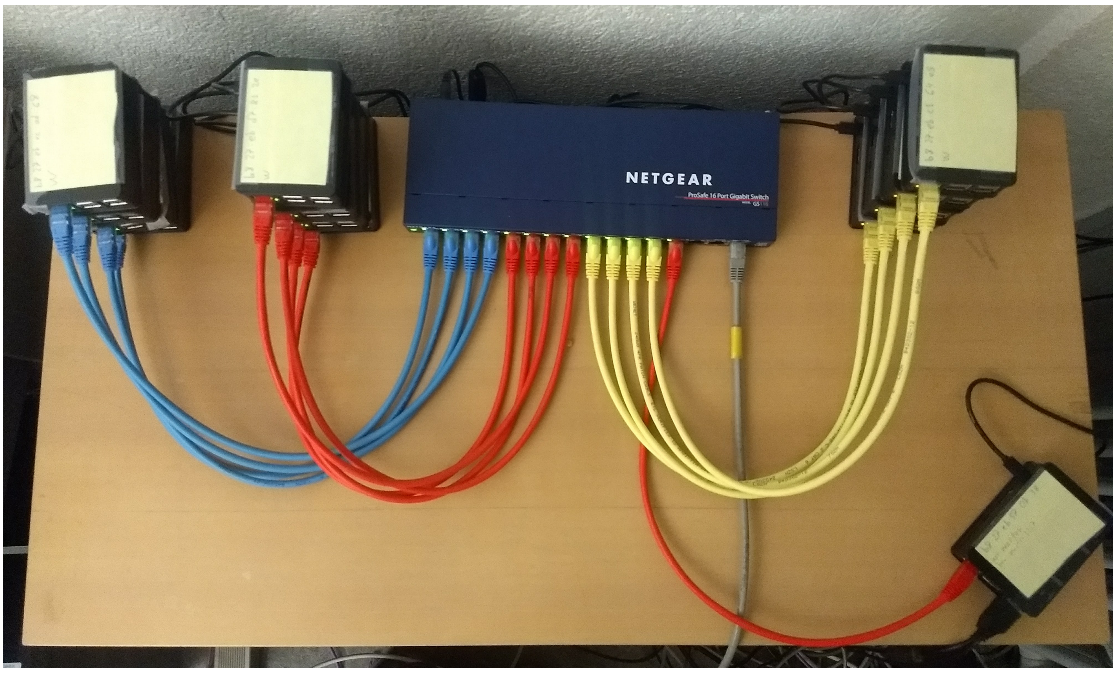


Figure 5.5: A Spark cluster of 13 Raspberry Pis. Master node (bottom right) and worker nodes connected via an ethernet switch.

this cluster is an SBC equipped with a quad-core processor at 1.2GHz and 1GB of RAM, and with a retail price below $40 USD it represents the lower end of crowd devices suitable for the CrowdCloud. The initial testing of the Spark computing framework on the Raspberry Pi cluster showed that crowdsourced computing with resource-constrained devices is feasible.

5.4 Conclusion

In this chapter we formalized the newly emerged architectural paradigm of crowdsourced systems, in the field of IoT. Crowdsourced systems are systems whose infrastructure is heavily relying on resources pooled from the general public, via crowdsourcing methodologies. Drawing a parallel from the requirements of a "wise" crowd in traditional crowdsourcing, we identify the requirements that must be met for a crowdsourced system to be functional in the field of IoT. Moreover, we delimit the high level components that form the "canonical" IoT-related crowdsourced system, namely the crowd, the crowdsourcer and the crowdsourcing platform. The crowdsourcer defines the task and the corresponding incentives, which is disseminated to the crowd via the crowdsourcing platform, through which the task results return back to the crowdsourcer. The crowdsourcing platform is a compound entity enclosing data and resource management modules, as well as a task handler and a resource gateway. It is common for crowdsourced systems to use an external crowdsourcing platform which is provided in a Platform-as-a-Service fashion; that being said, all of the functionalities can also be implemented inherently in the system. Finally, we exemplify the formal definitions with a use case implementing a crowdsourced system for provisioning of cloud infrastructure. We utilize the cluster computing framework Apache Spark, and deploy it in a cluster of 13 Raspberry Pis.

6 Crowdsourced Edge: A Novel Networking Paradigm for the Collaborative Community

6.1 Introduction

According to latest projections, it is estimated that a staggering 50 billion devices will be connected to the internet by 2030 [119]. These levels of growth will inevitably lead to an escalation of resource congestion in centralised computing paradigms, such as cloud computing, which, no matter the scaling up of their infrastructure, will be burdened with additional network latency and subsequent drops in Quality of Service (QoS). Moreover, next generation networks requirements include support for high density and low latency, calling for novel communication technologies (such as the use of millimeter-wave frequencies) as well as for innovative network architectures.

The above circumstances have fostered the sprouting of the edge computing paradigm, where computational resources are strategically positioned at the network edge in order to perform locally computationally demanding tasks, instead of outsourcing them to the cloud. This helps alleviate latency inferred from the physical distance to the cloud centers; indicatively, in a proof-of-concept study measuring latency, the response time of the execution of a highly sophisticated face recognition algorithm was reduced from 900 to 169 ms by moving computation from the cloud to the edge [120]. Besides minimizing distance-induced latency, a second pillar motivating edge computing is context-awareness; mobile applications become "smarter" when they can perceive their environment and act inline with its variations, a process that is facilitated by moving computational resources closer to the data source. Applications in the field of wearable cognitive assistance or those implicating virtual or augmented reality are typical examples in that context, as they require constant tracking of the environment to enable accurate decision making in real time.

Consequently, in recent years, stimulated both by industry and academia, there have been significant efforts and in-depth studies to concretise the edge computing paradigm. The first instance can be traced to Cisco and the introduction of Fog Computing [121], in which a virtualised platform for networking and computing services is positioned within the cloud-to-things continuum. Similarly, first defined from Satya et al [122], Cloudlet Computing proposes a network of small data-centers-in-a-box (cloudlets) to act as an intermediate layer between user and cloud. Along the same line but from a different perspective, Multi-Access Edge Computing (MEC) driven from the 5G latency requirements [123] envisions the enhancement of the Radio Access Network (RAN) with edge infrastructure positioned at the base stations [124].

All of the above paradigms converge in the establishment of a powerful, static or semi-static, network of resource-rich edge nodes which however, may prove not suitable for more flexible and responsive systems. In the context of IoT for instance, Gartner identifies the shift to more unstructured, dynamic architectures as one of the crucial IoT future challenges [125]. Another concern regarding the dominating edge computing paradigms is that, despite the implementation differences, in each case the edge network is managed centrally and is either consumed for internal purposes, or provided to third parties for Over the Top (OTT) service provisioning. Under this light, we propose here a radically different paradigm for edge computing: the *Crowdsourced Edge*. The Crowdsourced Edge or Crowdsourced Edge Computing (CEC) paradigm consists in involving individual crowd peers, within a zone of local proximity, in a collaborative edge computing environment. A CEC network is formed in an ad-hoc way and follows a dynamic mesh architecture, in which all peers are equal and share network resources in a democratised fashion. Decentralized mechanisms for partitioning and aggregation dictate resource and data management, following the principles of crowdsourced systems [126]. Driven from the individual and societal needs [127, 128], CEC capitalises on the emerging fields of Do-It-Yourself (DIY) networking, community wireless networking and crowdsourced systems, which are already shaping an urban-technological future outside the commercial internet paradigm.

We begin this chapter by first providing some necessary background on edge computing in the following section. Then, we present the architecture of CEC, as well the main research challenges imposed, highlighting also the key benefits of the proposed solution. In order to concretise the CEC paradigm, we then detail a specific use case regarding a video-enhanced missing object search, before finally, concluding this work.

6.2 Background and Related Work

6.2.1 Edge Computing

Edge computing is a computing paradigm in which substantial computational and storage resources are positioned at the internet's edge, in close proximity to user devices. So far, there have been three dominant instantiations of the high-level paradigm of edge computing: Fog Computing, Multi-Access Edge Computing and Cloudlet Computing. Each one of them possesses unique characteristics and challenges, as detailed in numerous thorough studies [129–132]. In order to position our work in relation with the established groundwork, we provide here a short overview of the above-mentioned edge computing concretisations.

6.2.1.1 Fog Computing

Fog Computing is a term used in some cases interchangeably with edge computing, nevertheless “fog" implies an intermediate layer between edge and cloud. It was first coined by Cisco in the development of a highly virtualised platform providing computing, storage, and networking services between end-devices and traditional cloud computing data centers [133]. Fog architecture enables the creation of a hierarchical infrastructure, where tasks and processes are coordinated between several layers from the edge to the cloud in a bottom-up fashion (Fig. **??**). The entities belonging to the fog layer correspond to routers, access-points or other specialized devices which are typically located not further apart from end user-devices than some tens of meters. To tackle the heterogeneity that comes with the various devices, the fog network proposes the use of emerging techniques such as Network Function Virtualization (NFV) and Software-Defined Networking (SDN) to create a flexible network environment.

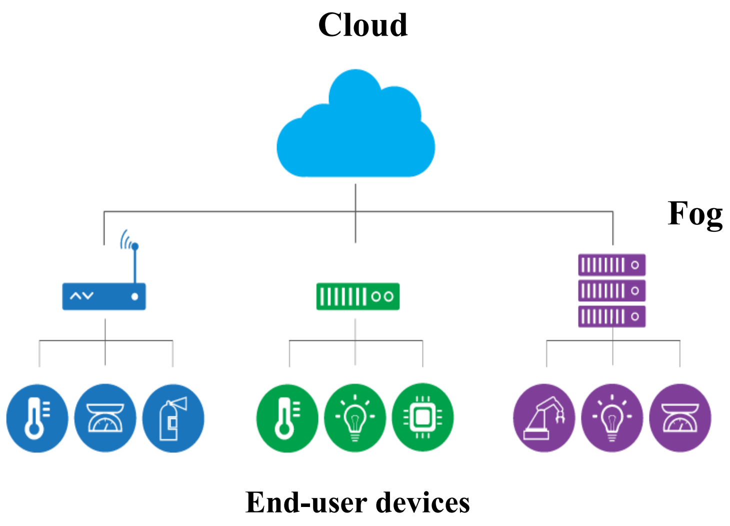


Figure 6.1: Fog computing layer architecture

A joint effort form academia and industry towards standardisation in fog computing started in 2015 with the OpenFog Consortium, with founding members from high tech industries such as Cisco and Intel, as well as academic institutions across the world including Princeton University. In 2017, the OpenFog Consortium issued a Reference Architecture, specifying high-level requirements from design to implementation, which was later declared an IEEE standard for fog computing [134].

6.2.1.2 Multi-Access Edge Computing

With the focus on mobile clients within the Radio Access Network (RAN), MEC proposes an edge computing implementation with edge servers at the RAN base stations [135] (Fig. 6.2). The first approach of implementing MEC can be traced in 2013 when Nokia and IBM introduced a platform allowing the execution of services within mobile base stations [136]. A more significant introduction came one year later though, when MEC was officially assigned an Industry Specification Group (ISG) as one of the key emerging technologies towards 5G, by the European Telecommunications Standards Institute (ETSI) [124]. At that time the "M" in MEC referred to "Mobile" edge computing, a term which was changed two years later to "Multi-Access" in order to broaden MEC's applicability to networks incorporating WiFi and other fixed access technologies. According to the ETSI white paper, MEC is characterized by five key features, namely on-premises, proximity, low latency, location awareness and network context information. Similarly to fog computing, MEC is closely coupled with the emerging technologies of SDN, NFC as well as network slicing, and there is substantial effort focused on the development of MEC specifications based on industry consensus. Indicatively, right now there are 68 members and 35 participants in the ETSI consortium for MEC, ranging from mobile operators, manufacturers, service providers as well as universities, which have the common goal of ensuring an open and interoperable MEC environment.

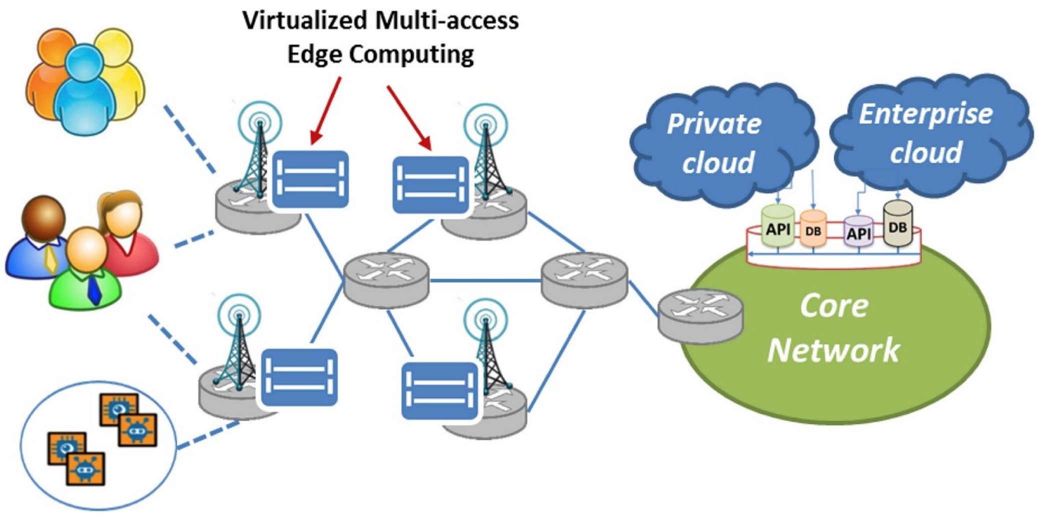


Figure 6.2: Multi-Access Edge Computing architecture [6]

6.2.1.3 Cloudlet Computing

Cloudlet Computing emerged as an evolution of Mobile Cloud Computing (MCC) which early in 2009 addressed issues of mobile delegation and task offloading but strictly with the cloud [137]. Cloudlets, firstly introduced by Satyanarayanan et al [138], are small resource-rich data centres that can be positioned strategically in close proximity to end-users, imitating the cloud and allowing for intense computations close to the data source (Fig. 6.3. More recently, Victor Bahl et al [139] introduced the -cloudlet equivalent- micro datacentre as an extension of today's hyperscale cloud datacentres, following the same architecture. The popular open source framework OpenStack for cloud computing has been customized to support cloudlet deployment, in the implementation OpenStack++ [140]. As it is imposed in all edge computing implementation, customized Virtual Machine (VM) hand-off mechanisms for cloudlets have been developed for dealing with mobile users, similarly to the way MEC handles RAN mobile subscribers.

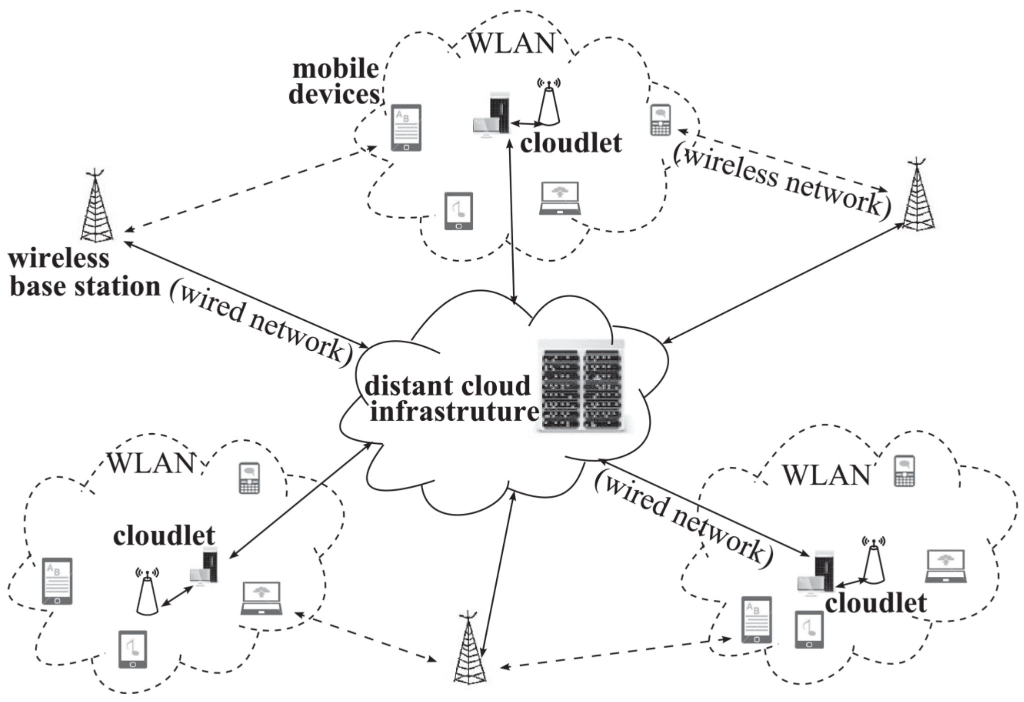


Figure 6.3: Cloudlet Computing architecture [7]

In the following section, we present the architecture and characteristics of a crowd-based edge computing, the Crowdsourced Edge. The Crowdsourced Edge is designed in line with the principles and guidelines of crowdsourced systems (detailed in the previous chapter) and, to this day, constitutes the first attempt to implement the edge computing paradigm as a crowdsourced system. Mobile Crowdsensing Systems (MCS) have been previously linked with edge computing and in particular with MEC [141, 142], but mainly as an additional architectural module providing either extra network coverage or human intelligence in task execution. We advocate that CEC possesses unique characteristics that differentiate it from the existing implementations and position it as a novel edge computing paradigm that drives the need for locally managed, context-aware applications, from a citizen/community perspective.

6.3 The Crowdsourced Edge

6.3.1 Architecture

The Crowdsourced Edge connects crowd peers belonging to a local proximity neighbourhood, in an edge-service provider network. The edge network is formed in an ad-hoc manner, either via traditional networking infrastructure already in place (LTE, internet) or by alternative local networking techniques like the ones developed in the emerging fields of DIY networking and offline networking [143]. A CEC network is formed by crowd peers connected in a dynamic mesh architecture, where each node depending on user preferences and the application context can take one or more of the following roles:

1. **Task handler.** Task partitioning and queuing is managed by the task handler. Nodes implementing the task handler are bound to have higher requirements in matters such as allocated memory or computational capacity, but also in performance robustness and overall reliability, given their key role in application workflow.

2. **Worker.** The worker nodes consume the tasks coming from the distributed task queue managed by the task handler. An abstraction mechanism is required to discover and expose worker resources in a homogeneous manner.

3. **Message broker.** Message brokers implement network routing functionalities including packet forwarding and maintaining a list of active routes, depending on the communication protocol.

We state here that the above distinction does not imply a delimitation in the physical sense, but it designates an abstraction model capturing the different system entities; the three categories reflect the high-level functionalities of the collaborative computing framework proposed in CEC. In fact, in order to adapt to

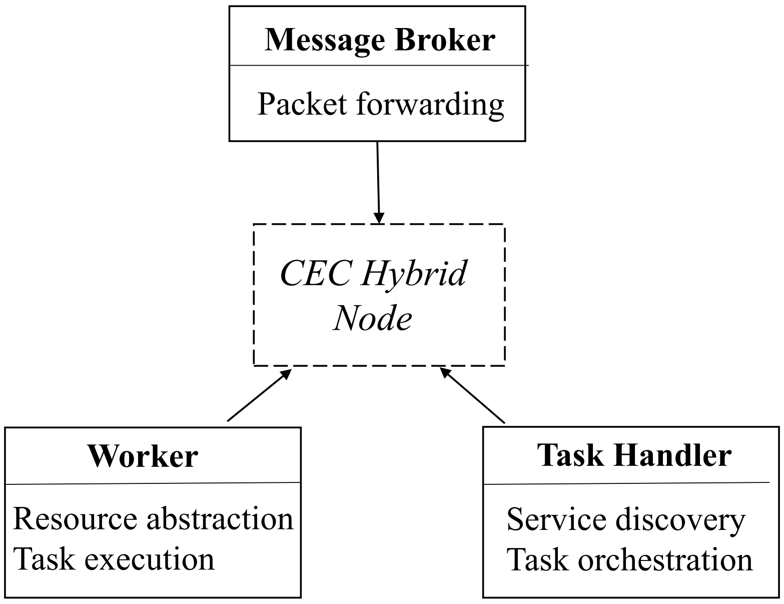


Figure 6.4: CEC node functionalities.

the dynamicity characterizing crowd environments, the library sets implementing the different roles should be packaged together in a single firmware equipping the *CEC Hybrid Node* (fig. 6.4), thus enabling for flexible and responsive management. As such, depending on application context, a CEC node can assume one or multiple of the above-mentioned system roles. This fully distributed architecture inherently addresses the functional requirements of "independence" and "decentralisation" outlined in IoT-related crowdsourced systems (see 5.3.2), since the system roles are not tied to specific nodes and can be migrated if needed. The requirements of "diversity" and "aggregation" reflect implementation challenges regarding general-purpose computing and resource management, which are discussed in detail in the next section.

Equipped with the libraries forming the CEC node, a crowd user can join the CEC network, and participate in applications using the pooled storage, computing, sensing or other resources in the network (Fig. 6.5). It is certain that, from the initial formation of the network to the successful exposure of crowd resources and distribution of edge services, a variety of challenges are raised, some identified

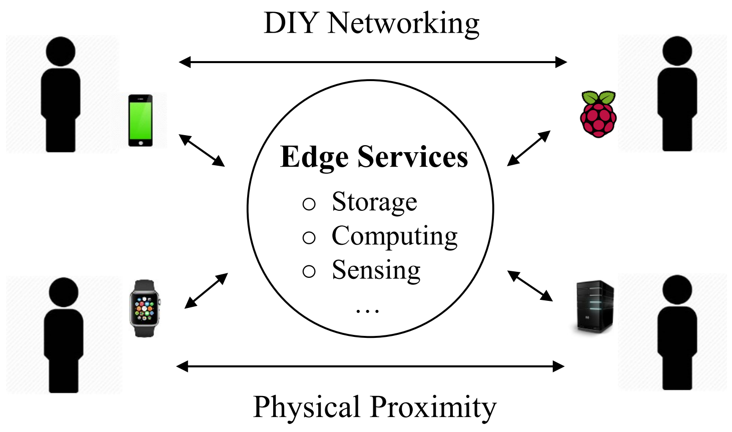


Figure 6.5: Crowdsourced edge high-level system architecture

as key challenges in "traditional" edge computing, such as node programmability and interoperability, and other that are specific to the CEC paradigm such as incentivization mechanisms and adaptation to a resource-constraint environment. Following, we identify the main challenges for computing at the Crowdsourced Edge, discussing available solutions and enabling technologies.

6.3.2 Research Challenges

Rapid formation of ad-hoc networks

Networking in a highly diverse, often underpopulated, environment is a very challenging task. In the easier scenario, a CEC network can be formed using existing networking infrastructure already in place, such as a building's WiFi APs or cellular network deployed by service providers. That being said, there are a lot of advantages from achieving an on the fly, ad-hoc, manner of forming the network such as added privacy from dis-intermediation and support for more application scenaria. Depending on the specific application, certain characteristics of the network may vary, e.g. the network lifetime for a concert event could range to some hours or the desired network coverage for indoor and outdoor settings may be substantially different. Elements from the emerging fields of DIY and offline networking, which leverage next generation mobile devices and SBCs to form ad-hoc networks quickly and efficiently, can aid the CEC network formation phase.

Flexible Routing

CEC networks, in principle, should support high dynamicity to respond to crowd participants being mobile and in general exhibiting unstable behaviour. Routing patterns that follow the publish-subscribe communication paradigm can fit well this setting, as nodes can request or advertise services in an asynchronous way. Lightweight implementations of the messaging protocols MQTT and AMPQ exist and can be used for implementing such a routing mechanism. In the lack of static nodes to act as message brokers, rendezvous routing algorithms which create direct paths between publisher and subscribers can be used, such as the one described in chapter 2, or else hybrid approaches like the one used in the Broccoli stack for distributed computing where all network nodes include message broker functionalities [144]. Alternatively, distributed routing techniques used in peer-to-peer (P2P) networking, such as distributed hash tables [145], are also good candidates for implementing a routing mechanism suitable for CEC networks.

General-Purpose Computing

Despite the advancement of general-purpose computing infrastructure, partitioning computing tasks to be executed in highly heterogeneous devices, both in terms of hardware as well as operating system, remains a daunting task. The use of Virtual Machine (VM) provisioning helps overcome this heterogeneity but VMs can prove too resource-demanding for user devices in the lower tiers. An alternative to VM provisioning is container technology, the most popular instance being Docker [146], which can provide fast deployment, small footprint and according to recent studies good performance in constrained environments [147], making it a potentially viable solution for CEC.

Data Consistency

The nodes in the CEC network may need to share information with other nodes. In a loosely coupled distributed system such as the one targeted by the CEC framework, ensuring data consistency while simultaneously maximising performance by allowing the nodes to operate independently is a challenge. Several consistency models have been developed in the distributed systems community which offer trade-offs between stronger consistency (correctness of data) versus performance [148]. On the one extreme we have strong consistency models which behave as if a single centralised node handles all the operations. This model requires synchronisation and significantly slows down the computation [148]. The other extreme is the weak consistency which does not provide any guarantees [149]. Other consistency models include Causal Consistency [149], eventual consistency [149] which lie somewhere in between the extremes.

Resource Management

Resource management in heterogeneous environments call for an abstraction layer to handle the diverse node resources, in order to enable higher level functionalities such as service discovery and queuing. An approach following the REST architectural principles should be followed in CEC APIs implementing those functionalities, in order to promote interoperability. Several fully-fledged framework for distributed computing have been developed (Hadoop, Spark, GridGain and more) which include a resource management platform, but very few of them are a good fit for a resource-constrained environment; Apache Edgent [150] and DC4CD [151] are two recent projects targeting resource-constrained devices but both are still in an incubating stage.

Data Privacy and Security

Data privacy and security in digital networks is a major concern in modern society, given a series of major data-breach cases of very large scale [152, 153]. The Crowdsourced Edge has to also address such concerns since edge services may also handle sensitive user data or require delicate hardware access depending on the application. Privacy-by-design [154] techniques such as decoupling of user profiles with sets of data in application development can solve this issue. Moreover, access control services such as authentication, authorisation and accountability can provide secure links between edge nodes. Given the distributed nature of CEC, something to also consider is the use of distributed ledger techniques, such as the blockchain, which can help enhance security with mechanisms like content validation and decentralised transaction.

Crowd Incentives and QoS

One of the tougher challenges for a collaborative P2P framework like the one proposed in CEC, is ensuring an acceptable QoS threshold. To achieve adequate QoS in crowdsourced systems, a critical mass of active crowd participants must be maintained; therefore individual peers must be a)incentivised to join the CEC network and b)have their expectations met in terms of Quality of Experience (QoE) in order to remain active and maintain a stable behaviour. Crowd incentivisation is a topic thoroughly investigated in crowdsourced systems, where schemes that reward participation and engagement are often proposed [155]. In CEC we envision a direct reward for the peer coming from utilising the edge services (extrinsic incentive) as well as a participation reward in the form of social acknowledgment of contributing for the common well (intrinsic incentive). In order to secure a decent level of QoE for crowd peers, fairness in terms of ensuring equal resource sharing and load balancing is essential. Opportunistic approaches in distributed computing often lead to a small number of nodes carrying the majority of the processing burden, while a large number of nodes contributing little to the working of the solution. This can not be the case for CEC as excessive single resource commitment may lead to issues like battery drainage and hardware stress which may encourage peers to drop the network. Finally, fault-tolerant database schemes such as the Hadoop Distribute File System (HDFS) or Resilient Distributed Datasets (RDDs) will be needed to avoid loss of information due to sudden node dropout, which would cause a plunge in QoE metrics.

6.3.3 Comparison with Existing Edge Computing Paradigms

As seen in the previous section, the Crowdsourced Edge inherits the intrinsic characteristics and challenges of edge computing, such as its decentralised nature and the need for abstraction and virtualisation which comes with managing diverse resources. Compared with the three established paradigms, detailed in section 6.2, CEC is consonant with the fundamental edge computing traits like e.g. the decentralised network architecture or the support for mobile users, nevertheless it differs fundamentally in aspects like the motivational force and the return value distribution.

In table 6.1 we present a detailed comparison of CEC with MEC, fog computing and cloudlet computing. We highlight a major difference that lies in the edge network ownership, which for the case of CEC is shared among the owners of the nodes forming the network, in a community-like manner, while for the other paradigms it resides with some private entity. This is coupled with a shift in the driving force from business-related, which is mostly the case in the established paradigms, to individual or community wishes. Another important variation in CEC comes from the dynamic nature of the proposed network architecture, which makes it highly suitable for short-term application scenaria such as art exhibitions or sport events. At the same time this brings additional degrees of complexity in implementation issues not to mention QoS; moreover, in the same respect, the processing capacity offered by the powerful edge-servers proposed in other paradigms are difficult to achieve in CEC. That being said, one can envision a highly scalable system allowing

	MEC	**Fog**	**Cloudlet**	**Crowdsourced Edge**
Ownership	Telco companies	Private entities		Local communities, individuals
Edge nodes	Servers in base stations	Routers, access points, gateways	Datacentre in a box	End-user devices (PCs, Smartphones, SBC)
Network architecture	Static, decentralised			Dynamic, distributed
Support for mobility	Yes			Yes
Processing capacity	High	Average	High	Limited
Cloud use	No	Yes		No
Driving force	5G enabler	Business, B2B	Institution	Community

Table 6.1: Comparison of Crowdsourced Edge Computing with the dominant edge computing paradigms.

the cooperation of virtually limitless users and, taking into account the increasing capabilities of mobile devices, high computational capacity may be feasible.

6.3.4 Key Benefits

The adoption of the spontaneous collaborative networks enabled by the Crowdsourced Edge comes with diverse benefits for the individual and the community. In addition, one could also foresee the sprout of novel market opportunities within this emerging digital ecosystem. Here we list the key points of expected benefits originating from the Crowdsourced Edge:

- *Vendor-neutral edge services.* The Crowdsourced Edge paradigm brings a truly open and accessible to all edge computing solution. This is reflected in the minimal hardware requirements allowing low-power constrained devices at affordable prices into play, while on the contrary we observe e.g. Cisco fog computing hardware cost ranging around several hundreds of US dollars. Likewise, cloudlets have been appearing in pay-per-use business models [156] while MEC infrastructure is intrinsically part of the telecommunication service-provisioning market. The Crowdsourced Edge, therefore, provides an alternative to the above with its vendor-neutral approach, placing power to the citizen and promoting self-sustainable digital communities.

- *Privacy via dis-intermediation.* As previously mentioned, data privacy is a major concern for next-generation networks. From the communication perspective, the risk of a breach occurring grows proportionally with the number of intermediaries involved. In this respect, CEC networks, which follow the principles of DIY networking and community or offline networks that are dis-intermediated from centrally managed service-provisioning entities, provide a more secure, privacy-preserving, environment for users to trust their data.

- *Data ownership and market opportunities.* Going a step further regarding user data security in the Crowdsourced Edge, data transactions can be uniquely linked to the users producing them with the help of blockchain technology. This digital proof of data ownership supports users in matters related e.g. to intellectual property, but it could also tinder a novel market ecosystem where users can trade their data using blockchain transactions.

- *Digital literacy.* Along with other emerging DIY technologies, the Crowdsourced Edge kindles a multifaceted engagement of the society with future technologies, paving the way for proactive citizens in digital smart communities. CEC platforms can help citizens to enhance their digital literacy by providing means of easy access to next generation networks, via tools available to all.

6.4 A CEC Use Case: Video Enhanced Object Search

In order to demonstrate the functionality of the Crowdsourced Edge we present here a prototype use case. The proposed application utilises the edge network to crowdsource a video-enhanced object search in a wide local area (e.g. a university campus), for the purpose of generating informative heat maps or identifying a specific object (f.i. a lost object). Leveraging the user devices' cameras and computational resources, all data processing is performed locally at the network edge, and only metadata, such as classifier parameters or object detection results, are transferred through the network.

Before detailing the application workflow and architecture, we go over two main design considerations in application development: i) the choice of classifier for object detection given the specific context constraints (e.g. energy, computational resources) and ii) the preservation of privacy when video recording and editing. We evaluate the more suitable methods and algorithms with respect to the above and subsequently we detail the overall application workflow with the help of two example scenarios.

6.4.1 Lightweight Object Detection

Deep learning techniques have grown to become the standard approach in computer vision tasks such as object detection, ever since AlexNet [157] won the ImageNet Challenge using deep Convolutional Neural Networks (CNNs). Nevertheless, the high accuracy achieved by CNNs usually comes with the price of high computational cost, which constitutes an impasse when it comes to resource-constraint application environment. Recent efforts attempt to optimise the number of operations in CNN classifiers by replacing the classical convolution with custom versions, for instance by using sequences of depthwise or spatial separable convolutions to achieve the same results. This has produced some significantly more efficient CNN architectures (Inception [158], NasNet-Mobile [159], MobileNet [160], Squeezenet [161], ShuffleNet [162] and more) both in terms of size and computational cost.

For the purposes of the CEC use case, the use of TensorFlow Lite [163] is envisioned, which is a lightweight version of the popular software library TensorFlow, tailored for mobile and embedded devices. TensorFlow Lite provides a rich set of machine learning APIs including the latest versions of lightweight classifiers used in computer vision, achieving impressive results in terms of computational efficiency on the latest smartphones, as can be seen in Table 6.2. In addition, the TensorFlow Lite binary is smaller than 300KB, with all supported operators linked, and less than 200KB when using only the operators needed for Inception and MobileNet [163].

Model	**Device**	**CPU**	**GPU**
Mobilenet_1.0	Pixel 4 / iPhone XS	23.9 / 14.8 ms	6.45 / 3.4 ms
SqueezeNet	Pixel 4 / iPhone XS	23.9 / 21.1 ms	11.1 / 15.5 ms
Inception_V2	Pixel 4 / iPhone XS	272.6 / 261.1 ms	87.2 / 45.7 ms
Inception_V4	Pixel 4 / iPhone XS	324.1 / 309 ms	97.6 / 54.4 ms

Table 6.2: Performance benchmarks of TensorFlow Lite classifier models on market leading mobile devices [8].

6.4.2 Privacy Aspects

Since this application prompts users to record video, and then this video is subdue to some processing, privacy aspects must be addressed to be in accordance with respective regulations. The exact legislation regarding video recording differs from one place to another and is often ambiguous, but in most cases the law dictates that when residing in outdoor public spaces, in which one is legally present, a person has the right to capture any image that is in plain view. When located on private property however, it is the property owner who sets the rules about the taking of photographs or videos. Therefore, in order to ensure proper usage of this application, all of the above will be conveyed to the users in the form of a legally binding privacy policy upon application installation. The exact clauses may vary depending on the country or state location. Regulations are typically more sensitive regarding audio recording, which is why the app will restrain audio capture when video recording. When it comes to video editing, the main privacy-threatening concerns raised regard the case of face recognition, for the protection of which a variety of techniques (such as video redaction and denaturing) have been developed. In our case though, face recognition is not envisioned at the moment, and in addition, the actual video files never leave the device in which they were recorded, since all processing happens at the edge. This adds an extra layer of privacy to the application, since it is harder for a potential attacker to retrieve sensitive information solely from transmitted metadata.

6.4.3 Workflow and Architecture

The functionality of the proposed application is twofold: the first operation mode allows the generation of heat maps, based on a specified area and generic object; the second launches a search for a specific object (e.g. a lost object). In Table 6.3 we detail two example scenarios illustrating the different operation modes. In both cases, the main mechanism remains the same, which is the crowdsourced object detection in video files capturing the environment of each user. What changes in the specific object search is that first the app performs an object-specific feature extraction, based on user-provided pictures, with the aim of gathering identifying characteristics of the object and translating them to quantifiable parameters in the object detection alogrithm. Besides obvious identifying traits such as color or shape, deeper-level latent features can also be extracted as the input dataset grows. When requesting heat maps of generic objects on the other hand, the set of pre-trained models included in TensorFlow Lite can be used directly and this step can be avoided.

Scenario A: Informative heat map generation
1. It is a crowded weekend day due to a sports event on the university campus. A student is looking for a calm outdoor place to sit and read. 2. He/she opens the app and requests a heat map of the object "human" in the campus area. 3. Crowd users in campus proximity receive a notification to scan their environment with their camera. Object detection of "humans" is performed in each edge device upon recording. 4. Results from the dispersed scan are returned on the student's device and the app generates the requested heat map, where the less crowded areas can be easily spotted.
Scenario B: Search for missing object
1. A person realises their bike is missing. He/she feeds the app with past pictures of it (or market ones if he/she doesn't have any) and requests a specific object search. 2. The app extracts specific object features for this bike to be used for object detection, and pushes them to the edge network. 3. Upon reception of the object detection task, other users with camera-equipped devices (smartphones or laptops) join the search by scanning their surroundings. 4. When a possible match is found, a private thread opens between the two users involved.

Table 6.3: Demonstrative scenarios.

This mobile application is designed for Android OS, which is open source and currently possesses more than 74% of the mobile operating system market share worldwide [164]. The architecture (figure 3.6) denotes the two main utility blocks which is the object detection module and the heat map generator, alongside the CEC APIs and an interfacing control layer. The CEC APIs enable the device to connect and interact with the CEC network, including functionalities such as retrieving network routes, preparing outgoing messages, broadcasting, parsing incoming messages etc. The object detection module encapsules the TensorFlow Lite library and finally the heat map generator is implemented in a straightforward way via the google maps heatmap utilities [165].

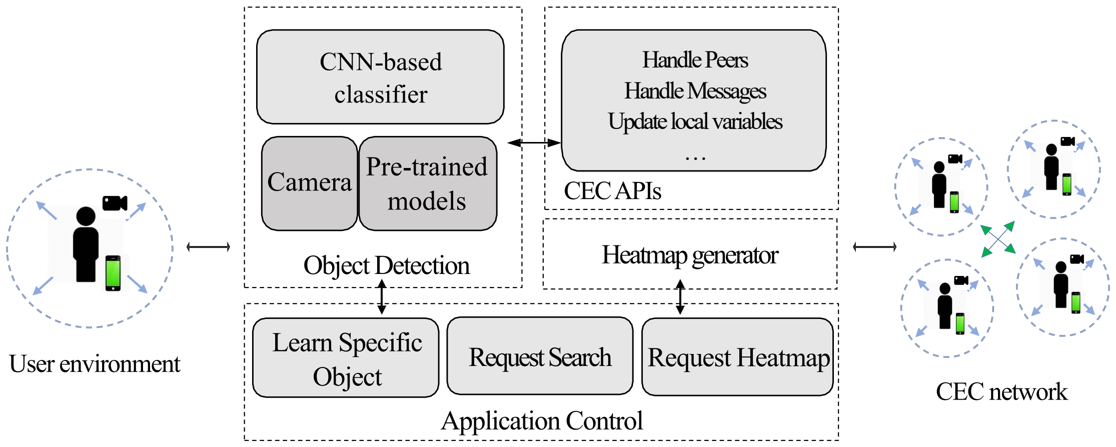


Figure 6.6: Application architecture.

6.5 Conclusion

In this chapter we presented the Crowdsourced Edge, a novel networking paradigm connecting individuals belonging to a zone of local proximity. The crowdsourced edge suggests the convergence of two emerging topics, those of edge computing and crowdsourced systems. Edge computing emerged in response to the increasing demand for low latency and context awareness in mobile computing. All edge computing paradigms though, incline to rigid deployments of substantial infrastructure linked with centrally-managed entities. To that end, we demonstrated how the Crowdsourced Edge employs more widely accessible mobile devices, to provide a user-centered edge computing framework, uncoupled from intermediaries. We detailed the paradigm architecture, in which CEC hybrid nodes are connected in a dynamic mesh topology. In order to support more flexible and dynamic systems, CEC nodes, depending on application context, encompass one or more of three abstracted functional system layers: a)worker; b)task handler; and c)message broker. This architecture differentiates the Crowdsourced Edge from the existing edge computing paradigms, as can bee seen in table 6.1, in terms of network ownership but also in the driving force.

Crowdsourced systems have found fertile ground in the field of IoT, enabled by every-day mobile devices possessing increased capabilities and SBCs which are becoming abundant. Formal requirements have been identified regarding such systems; we demonstrated how, thanks to its decentralized, responsive architecture, the Crowdsourced Edge meets these functional requirements. The CEC paradigm is undoubtedly faced to a series of research challenges, ranging from networking aspects to computing and resource management, which we detail in sec. 6.3.2. Finally, in section 6.4 we presented an ongoing CEC use case which involves crowd-users in a wide area, video-enhanced, object search.

7 Ad Hoc Formation of Offline Networks for Crowdsourcing using Non-Rooted Smartphones

7.1 Introduction

The emerging field of Do-It-Yourself (DIY) networking already speaks to us about hybrid urban spaces which, enabled by the evolution of networking technologies, empower citizens towards collaboration and collective awareness in a way never before possible [166]. The benefits from such a leap for the individual are many, such as digital services of enhanced privacy via dis-intermediation and data ownership preservation, but the challenges that come with it are also significant. In the previous chapter, we defined the paradigm of the Crowdsourced Edge, which along the the same line attempts to realize edge computing as a crowdsourced system, and we detailed the main challenges of such an undertake.

In this chapter, we focus primarily on the networking aspects in the building of a crowdsourced computing framework. The objective is to develop a method to efficiently connect as many crowd peers as possible in local proximity, utilizing solely the crowdsourced resources to maintain the network and manage the crowdsourced data transactions. All functionalities should be fully distributed, as individual peers may exhibit unstable behavior and this should not jeopardize the validity of the solution. These design considerations have led us to consider the field of Mobile Ad-hoc Networks (MANETs), which are designed not to rely on fixed infrastructure and which deal with a network environment dictated by high node mobility.

In particular, we focus on a special case of MANET, one which is comprised solely of smartphones, sometimes referred to as a Smartphone Ad Hoc Network (SPAN) [167]. This is because smartphone users worldwide today have surpassed three billion, while forecasts of growth for the next few years are in the several hundreds of millions [168]. This is important because it greatly favors crowd engagement which is paramount for crowdsourced systems, as people already carry on them, or in their close proximity, an increasingly powerful computing device with significant networking capabilities that is the average modern smartphone.

A major design consideration for the formation of SPANs is the choice of wireless networking medium; from the family of protocols based on Bluetooth and WiFi, we choose to utilize WiFi Direct, for its high bandwidth as well as its high rate of compatibility with off-the-shelf smartphones. Using WiFi Direct, Peer-to-peer (P2P) networked groups can be formed similarly to traditional WiFi networks, with one device acting as a dedicated Access Point. This topology, however, is limiting for MANETs as it bounds the formed network to single-hop communication patterns. To overcome the above issue, we introduce a custom heuristic employed in the application layer, based on a method found in the literature.

Apart from the communication mechanism, another design consideration is the routing protocol employed in the formed network. Inspired by reactive protocols in MANETs, which build routes on-demand, as well as data collection protocols used in IoT, we present a protocol adapted for the purposes of our crowdsourced application framework. Our protocol follows the pattern of outgoing requests for data/services and asynchronous collection of results, based on route piggybacking and controlled flooding. Finally, we develop a mobile application for Android OS, which utilizes WiFi Direct and employs the mechanisms discussed above to provide a crowdsourcing P2P framework. The prototype application is tested with 4 market smartphones, arranged in two different topologies (star and chain) and a crowdsourced sensing task.

The rest of the chapter is structured as follows: first, we provide some background on related work and enabling technologies in section II. Then, in section III, we detail the methodology regarding the network formation using WiFi Direct, as well as the proposed routing protocol. Then, in section IV, we perform a proof-of-concept evaluation via a prototype use case using four different smartphones. Finally, section V concludes this work.

7.2 Background

7.2.1 Wireless Networking with Smartphones

Aside from the centralized LTE network provided by telecommunication service providers, there are two main wireless networking standards which can be used for creating P2P connections between smartphones: Bluetooth and WiFi. In the following we go over their main characteristics and compare them as a choice of networking standard for our application.

Bluetooth

Bluetooth, developed in the late 1990s, is a wireless standard used for transmitting data over short distances. Its recent evolution Bluetooth Low Energy (BLE) achieves lower energy consumption and comes with a lower cost compared to its predecessor, while maintaining a similar communication range. BLE has been widely used in IoT applications, especially due to its low cost and small overhead. Since it was designed for transferring lightweight data such as sensors readings, it provides a limited bandwidth around 1 Mbps. In terms of communication range, it is largely dependent on the obstacles so it may vary, but it is typically in the order of some 10s of meters. Finally, concerning security, Bluetooth has been linked with serious security flaws in the past but most of them were solved with the newest standards, nevertheless the latest security standards may not be implemented.

Regarding the communication pattern, Bluetooth was initially designed to enable point-to-point connections, assuming a star network topology. The need for multi-hop communications in IoT applications though, lead to Bluetooth Mesh, a recent version based on BLE who allows many-to-many patterns using a flood network principle [169]. Nonetheless, mesh topology support has increased the vulnerability of BLE to security threats and there is still a lack of detailed studies related to these new security issues [170]. Moreover, the Bluetooth Mesh standard is not yet supported on smartphones on the market, which is where our application is targeted at.

WiFi

WiFi is a technology enabling radio wireless local area networking based on the IEEE 802.11 standards. It is significantly faster than Bluetooth, attaining bandwidth levels up to 10 Mbps, and has also a larger communication range reaching almost 100 meters in outdoor settings. These advantages come with a price though as WiFi hardware is more expensive and its power consumption is considerably higher compared to BLE. That being said, Bluetooth is also less secure than WiFi. for which specialized security protocols (WEP, WPA) have been developed and thoroughly tested.

There are two operation modes allowing for different networking topologies defined in the WiFi protocol: a)infrastructure mode and b)ad-hoc mode. Infrastructure WiFi employs some special nodes, called Access Points (AP), which support one-to-many connections and in the general case act as a bridge with wired network infrastructure and usually the internet. Other nodes of the network first connect to the AP and every further connection (with other nodes, the web) is relayed via it. On the contrary, ad-hoc mode allows for direct connections between WiFi nodes without the need of AP intermediation. It is also referred to as Independent

Basic Service Set (IBSS) or a peer-to-peer mode. When it comes to smartphones, WiFi ad-hoc mode is still quite underdeveloped as it is not inherently supported by the dominant smartphone operating systems. It can be used in selected devices after rooting, which allows to make certain modifications in the system files, kernel or drivers, but this process is difficult and far from being user-friendly. Nevertheless, a project designed by the Homeland Security System Engineering and Development Institute utilizes the WiFi ad-hoc mode to form SPANs in the context of natural disasters or terrorist incidents where existing network infrastructure is overloaded, destroyed, or compromised [171]. Even though the prototype network implementation is successful in that project, it is doubtful that in a real world scenario crowd participation would reach sufficient levels because of the limitations mentioned above.

WiFi Direct is another WiFi variant similar to the WiFi ad-hoc mode, in the sense that it allows device-to-device communication without the use of APs, but with certain structural differences. In fact, in WiFi Direct devices negotiate with each other to decide which node will act as an AP; this in essence simulates a WiFi network with a "soft" AP, with all communication going through it. That way, a Wi-Fi Direct device can concurrently maintain a connection to other Wi-Fi Direct devices, as well as an infrastructure network, which is not possible with Wi-Fi ad-hoc mode. The most important advantage of WiFi Direct, however, is that it is far more widespread; the WiFi Alliance to this day has certified more than 550 products supporting it, and the Android OS provides open-source APIs for developers to access WiFi Direct on non-rooted smart phones. Nevertheless, WiFi Direct's restrained topology of static device-to-device networks makes its use for dynamic ad-hoc network difficult. The newest WiFi extension, called WiFi Aware, is very promising on that front since it provisions the forming of P2P connections without a centralized coordinator, in a fully ad-hoc manner. Since it is still in an infantile stage though, and very few smartphones on the market support it, so it is not a good fit for this work.

Given the trade-off between the different networking standards, we choose for this SPAN implementation to go with WiFi Direct. Bluetooth's limited bandwidth and shorter communication range are considered as major drawbacks for this application, with the added value from low power consumption not quite compensating, given also the increased power capacity of modern smartphones. Then, from the different WiFi versions, WiFi Direct fits best our intentions, aligning with the design goal for a highly compatible solution in order to favor participation. In order to overcome WiFi Direct limitations, we introduce a custom modification, implemented in the application layer, that enables multi-hop routing. The method is detailed in the methodology section.

7.2.2 Routing in Mobile Ad Hoc Networks

Mobile Ad Hoc Networks (MANETs), evolved from the early Packet Radio Networks (PRNets) [172] initially funded by the US Defense Advanced Research Project Agency (DARPA) for military applications, form a specific category of wireless networks. Designed to operate in unstable environments, which are dictated by high node mobility, they are characterised by a continuous self-configuration and self-organising of their resources, enabled by total decentralisation of networking tasks. Not relying on a fixed infrastructure, MANETs have been often employed in applications providing emergency responses in natural or man-made disasters, where original networking infrastructure is compromised. Another range of applications of MANETs lies in data gathering in unstable or inhospitable environments, where traditional networks cannot be deployed.

Developing efficient routing protocols for MANETs is a challenging task and has been an active area of research area during the past years, with various proactive and reactive routing protocols being proposed [173]. A key issue to address is the necessity for the routing protocol to be able to respond rapidly to topological changes in the network. Prominent examples of reactive protocols used in MANETs are the Dynamic Source Routing (DSR) and the Ad-hoc On-demand Distance Vector (AODV) protocol, in which nodes discover routes on-demand in request-response cycles. When the regular maintenance of routing tables is afforded proactive protocols can be employed, with the Optimized Link State Routing Protocol (OLSR) and the Destination Sequenced Distance Vector Protocol (DSDV) being the most frequently used. For the proposed SPAN implementation, we examined mostly the reactive protocols, such as DSR, and also took inspiration from data collection protocols used in IoT, to come up with a custom protocol adjusted to fit the parameters of the problem. The exact mechanism is detailed in the methodology section.

7.3 Methodology

In this section we detail the methodology followed regarding two principal design considerations for this implementation, which are how to overcome the topological limitations of WiFi Direct, and the definition of the routing protocol.

7.3.1 Multi-hop Networks with WiFi Direct

The essential functionality of WiFi Direct is to enable WiFi devices to connect directly without first joining a central AP. WiFi Direct devices, also referred to as P2P Devices, communicate by establishing P2P Groups, which are functionally equivalent to the traditional WiFi infrastructure networks, with a "soft" AP being elected upon network formation. The device assuming the role of the AP in the P2P Group is referred to as the P2P Group Owner (GO), and the rest of devices act as clients and are referred to as P2P Group Members (GM). The choice of GO is dynamic and is negotiated at the time of initial network setup, based on intent and capabilities of each device. Initially, two P2P devices discover each other via advertising messages, similarly to the functionality of WiFi AP beacons. When a connection request is made by either device, they negotiate their roles (GO and GM) in order to establish a P2P Group. Once the P2P Group is established, other devices can join the group as clients (GMs), like in a traditional WiFi infrastructure

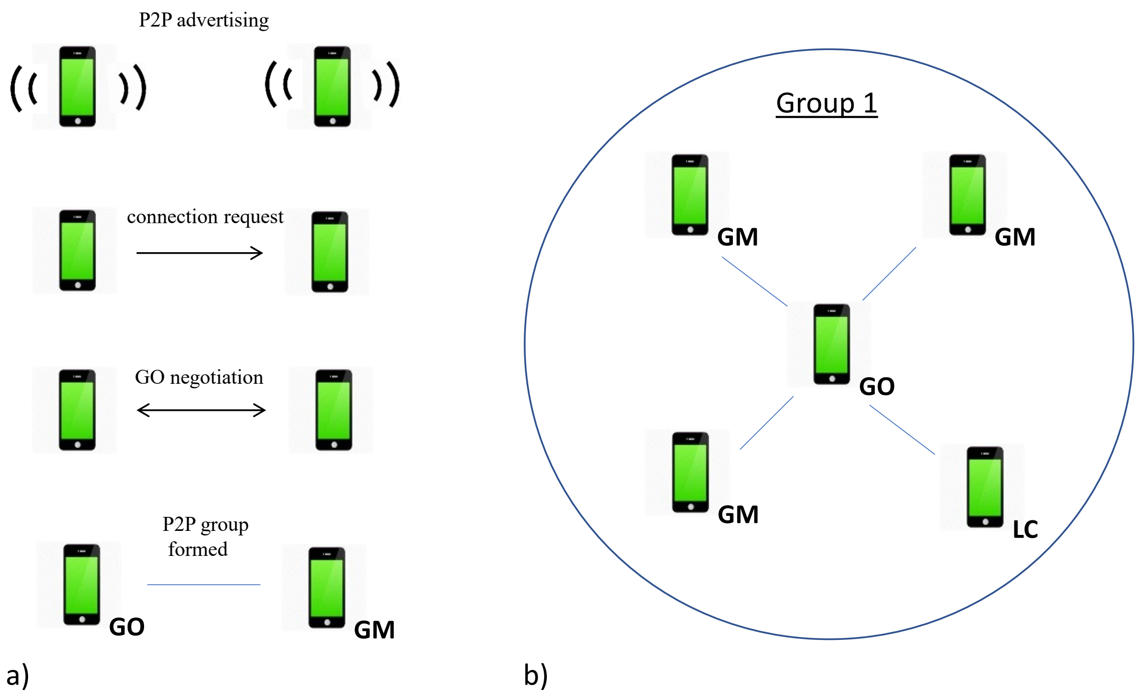


Figure 7.1: Formation of P2P groups with WiFi direct: (a) Stages during group formation (b) A formed P2P group with a GO and 3 GM as well as a Legacy Client (LC).

network. WiFi legacy clients (LC) can also connect with the P2P GO, as long as they support the required security mechanisms. Figure 7.1 depicts the stages of the P2P group formation.

The main limitation of the WiFi-Direct network structure comes from the fact that one device can only be either a GO or a GM at a time. In other words, each device can only belong to one group so if it wants to transmit a message to a device that is in range but belongs to another group, it has to either join the same group as the target device or it has to establish a new group with the target. This limits the WiFi Direct network to a one-hop ad-hoc network, with the GO being the gateway device. Theoretically, this could be overcome through multiple physical or virtual MAC entities, but both the multiple MAC functionalities and the simultaneous operation in multiple groups are out of scope of the standard. Figure 7.2 depicts two overlapping P2P groups, which are unable to transmit messages to one another, even though some of their nodes are in range.

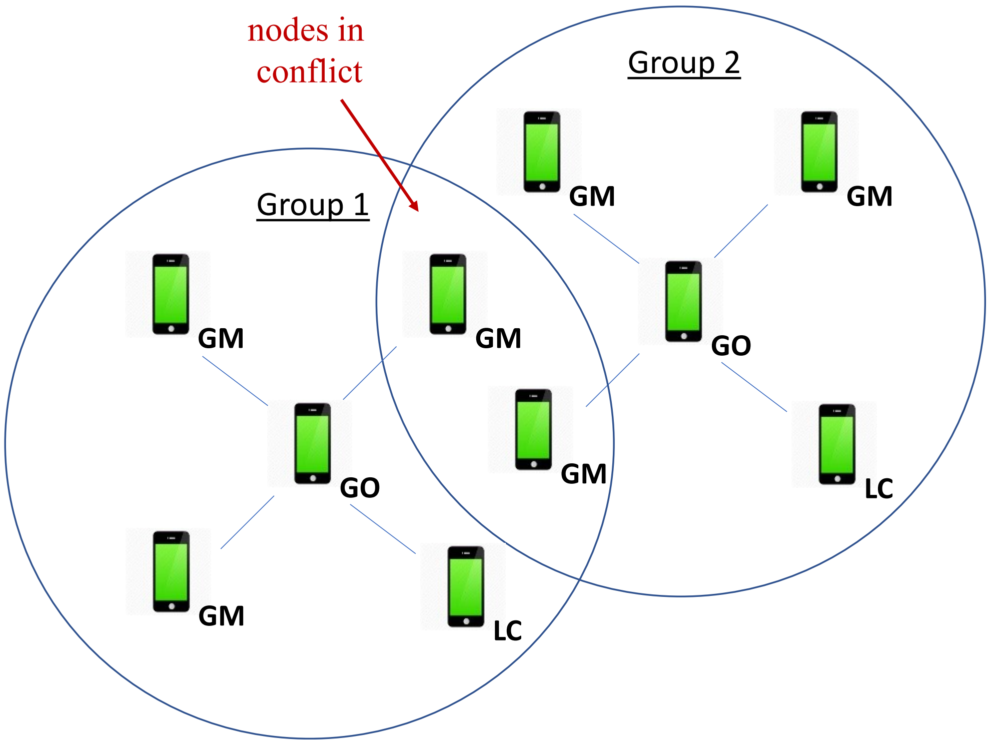


Figure 7.2: Two formed P2P groups. Nodes in the intersection are unable to relay messages in between groups due to WiFi Direct limitations.

Different ways have been proposed in the literature to surmount this issue, mostly by trying to create a custom gateway with the nodes in the groups intersection. For instance, Funai et al [174] propose the periodic switching of roles of the gateway nodes so as to simultaneously serve both groups. In a similar approach, Duan et al. [175] attempt to implement the gateway node by the GO from the first group to connect as a legacy client in the second group using the "WLAN" interface. However, both approaches introduce certain limitations on the flexibility and adaptability that should dictate a MANET. From a different perspective, the Mumble framework utilizes the WiFi Direct's Service Broadcast and Discovery (SBD) messages during the advertising phase, to transfer actual messages of content in an ad-hoc manner [176]. This method though has severe drawbacks in terms of bandwidth as SBD packets are strictly limited in size. In this work we follow an approach closer to the work of Liu et al. [177], where connections are established solely when a data message is to be transmitted. In particular, all nodes are kept in discovery mode, advertising their credentials in periodic beacons. Based on these advertisements, each node can deduct some information that can be helpful in message routing, which will discussed in detail later. Finally, when a node wants to pass some information then it requests a connection with the corresponding neighbor, and as soon as the data is transferred both nodes go back to the advertising mode. That way, a message can be relayed hop by hop to a distant network node, with the disadvantage of this method being more slow because of the repeated group formation stages. Nevertheless, we deem this approach a very good fit for our application, since we care about a fully distributed solution, not relying in single nodes, and relay speed is not so important. Figure 7.3 shows a step-by-step example of how this method operates in a message transaction at a network of 4 nodes.

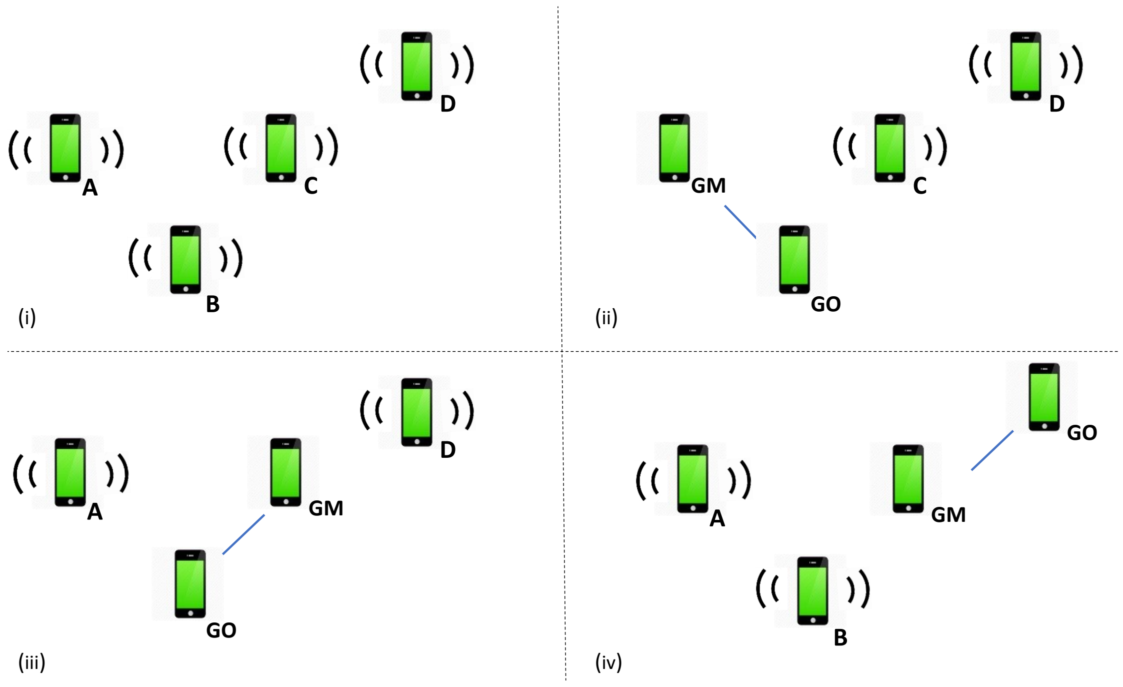


Figure 7.3: Example of a message relay from node A to node D using the proposed method.

7.3.2 Routing Protocol

We assume a network of n independent nodes, with the existence of an overlay network enabling point-to-point communication. Nodes are highly mobile, and may disappear and reappear at a later time. This reflects the behaviour of a crowd frequenting some specific area, e.g. a workspace or their apartment building. At any given time, every node possesses information only about its immediate neighbours (1-hop network knowledge). In this network setting, we want to implement a routing protocol enabling the following communication pattern: a single node requests some data from all other nodes in the network, and the other nodes reply with a data packet in an asynchronous manner.

Reactive routing fits best the problem description, as maintaining routing tables (proactive routing) comes with a low value to effort ratio in this setting. This is due to the high mobility of the nodes which make the added overhead from periodic beaconing messages not very fruitful. Combining elements for the reactive protocol DSR and the Directed Diffusion [17] protocol used in IoT, we propose for this work a data collection protocol which we call RE-COLLECT. Using three distinct message types, *REQUEST*, *COLLECT* and *RESPONSE*, the RE-COLLECT protocol functions as such:

1. When a node is looking for some data, it broadcasts a message of type *REQUEST*. At each message relay the traversed route is piggy-backed in the message.

2. Upon reception of a *REQUEST* message, each node proceeds to forward it to all its neighbours, unless they belong to the piggybacked route. Moreover, in the payload of a *REQUEST* message the task specifications are enclosed (implemented in the form of an executable in our prototype app), which are stored in the device by the enclosing application and may be executed instantly or at a later time. Once executed, the data is saved and await for collection.

3. The originator node initiates the data collection by broadcasting a message of type *COLLECT*, which is flooded in the network just like a REQUEST.
4. The difference is that upon reception of a *COLLECT* message, each node checks if the data is ready to be sent back. If yes, a *RESPONSE* message is sent back following the piggy-backed route backwards.
5. When backtracking at the piggy-backed route, if a node is no longer available the message is broadcasted to all neighbors instead.
6. The originator node can re-initiate a *COLLECT* dissemination at any time, to try to collect more data that may be available at a later time.

Since this protocol is targeting non-rooted smartphones, all routing logic was implemented in the application layer. In order to achieve the desired functionality a packet object was defined, emulating the networking functionalities as well as serving the application logic. As such, the defined packet was divided in the following fields (depicted also in Figure 7.8) :

- **ID**. The packet ID is a unique identifier used to distinguish each packet traversing the network
- **Time to Live (TTL)**. The TTL field contains a preset value which is diminished upon every relay, and once it reaches zero the packet is dropped.

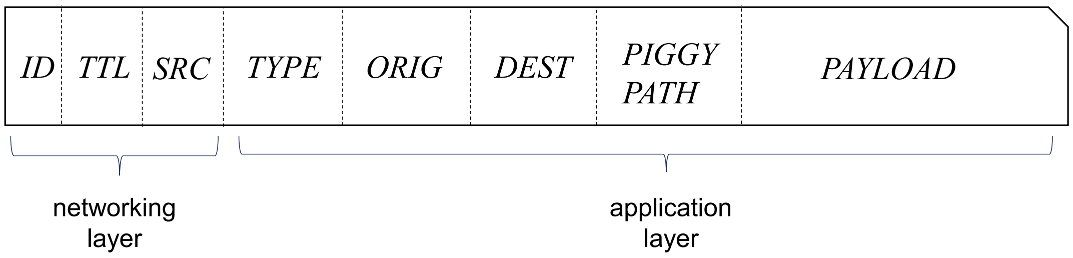


Figure 7.4: Packet fields of the RE-COLLECT packet.

- **Source (SRC)**. The SRC field contains the node id which transmits the packet every time.
- **TYPE**. The TYPE field contains a code indicating the message type, which can be *REQUEST*, *COLLECT* or *RESPONSE*.
- **Originator (ORIG)**. The ORIG field contains the node id of the node which initially requested the data.
- **Destination (DEST)**. The DEST field contains the node id of the destination node in *RESPONSE* messages. It is left empty for messages of type *REQUEST* and *COLLECT*.
- **Piggy-backed route (PIGGY PATH)**. At the PIGGY PATH field, the piggy-backed route is stored and updated every time.
- **PAYLOAD** At the PAYLOAD field, the task specifications are stored in REQUEST messages, as the data that are being collected in *RESPONSE* messages.

The above packet structure is exactly the one used in the Android application developed to experimentally evaluate this work, detailed in the following section. In fact, the 3 first fields correspond to some of the lower level networking aspects which normally would be implemented in the networking layer, while the rest of the fields correspond to the application logic and would normally be encapsulated in the packet payload. Since this is the way the packet was implemented in the Android application, we chose to present it here as such. That being said, the fields displayed corresponding to the application layer could be all residing inside a parent payload field.

From an algorithmic perspective, the protocol logic is encapsulated in the node functionality parsing an incoming message. In the following figure, we provide a simplified pseudocode for the above function.

```
Upon MSG reception
#Check if message is expired or has been already received
if (MSG.ID in received_ids) or (MSG.TTL==0) then
    ignore;
else
    #Check message type and act accordingly
    if (MSG.TYPE==REQUEST) then
        #Submit encapsulated task and propagate message
        submitTask(MSG.PAYLOAD.getTask());
        MSG.PIGGY_PATH.append(SELF);
        bcast(MSG);
    else if MSG.TYPE==COLLECT) then;
        #Propagate message and send back response if data is ready
        bcast(MSG);
        if DATA_READY then
            RESP = new MSG(DEST=MSG.ORIG, TYPE= RESPONSE,
            PIGGY_PATH=MSG.PIGGY_PATH, PAYLOAD=resp_data);
            ucast(RESP, MSG.PIGGY_PATH[last]);
        else
            ignore;
        end if
    else if (MSG.TYPE==RESPONSE) then
        #Check response destitation
        if (MSG.DEST==SELF) then
            dataArrived();
        else
            #If not for me forward using piggy_path
            MSG.PIGGY_PATH.remove[last];
            if (MSG.PIGGY_PATH[last] in neighbors) then
                ucast(MSG, node);
            else
                bcast(MSG);
            end if
        end if
    end if
end if
```

Figure 7.5: RE-COLLECT pseudocode upon message reception.

7.4 Experimental Evaluation

Android Application

In order to test experimentally the proposed method overcoming WiFi Direct limitations to build multi-hop networks (detailed in section 7.3.1), as well as the RE-COLLECT protocol, we developed a mobile application for Android OS. The app was developed in Android Studio and is compatible with smartphones supporting the Android API version 16 and above. Once the WiFi is enabled on a device, the peer discovery becomes active and the available peers (1-hop neighbors) appear in the app main screen (Figure 7.6). Two buttons are placed in the bottom of the screen which upon click trigger a REQUEST or a COLLECT message respectively, following the formula of the RE-COLLECT protocol. To coordinate the various message actions, a message queue class is employed that handles the outgoing messages, pushing them appropriately depending on the network state.

Since for every message transmission a P2P connection needs to be established, a delay interval is introduced between every message forwarding, in order to allow the devices to successfully reset their status to the peer discovery state. The exact delay value is an important factor which, as larger as it is selected, it favors the robustness of the protocol as we will see in the experiment run. Finally, to support the sharing of tasks, an abstract crowd task class was defined, containing a serializable executable which is to be programmed depending on the application, and that is inserted on the RE-COLLECT messages' payload. As such, when a REQUEST message containing a task is received, it is saved in a task registry and is displayed on the app "task overview" pane, from which its execution can be launched (Figure 7.7).

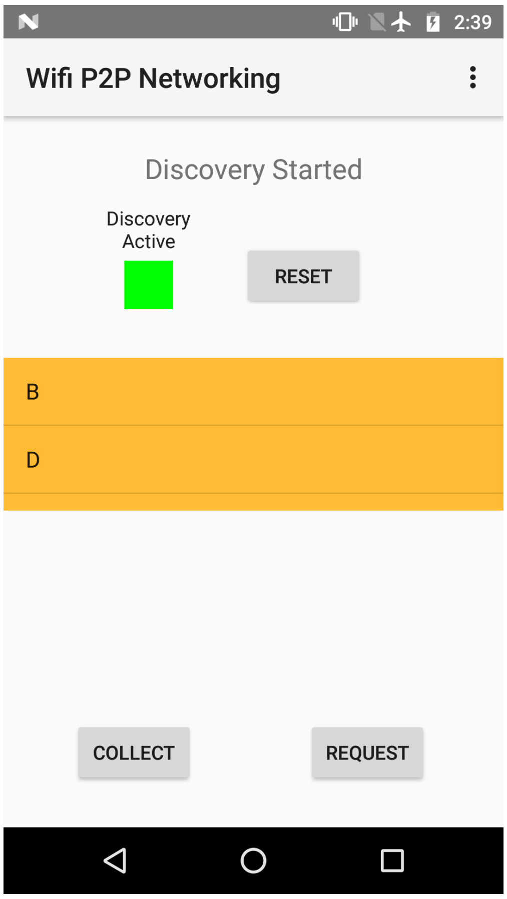
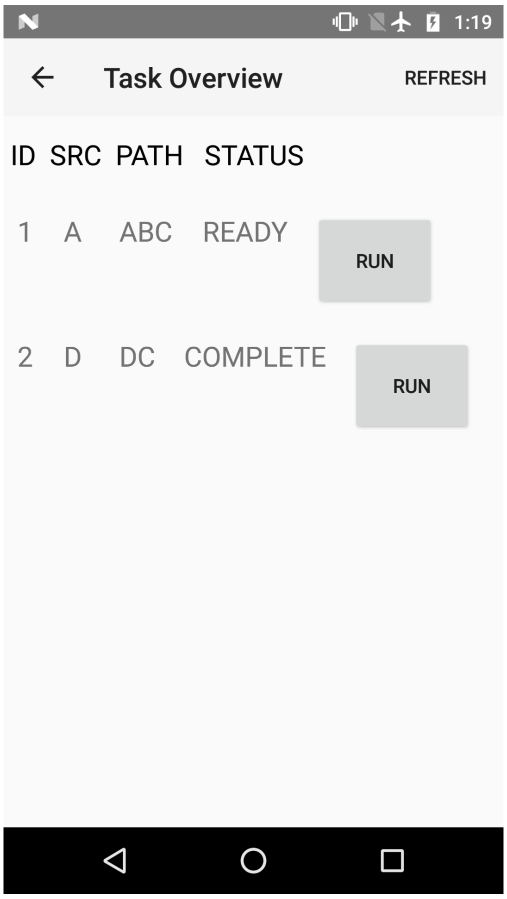


Figure 7.6: App home screen.



ID	SRC	PATH	STATUS
1	A	ABC	READY
2	D	DC	COMPLETE

Figure 7.7: Task overview tab.

Experiment Setup

To evaluate the efficiency of the proposed methods, a simple sensing task was created which polls a device's built-in illuminance sensor to capture a sensor reading. Four different Android phones were used for the experiment, their specifications being detailed in Table 7.1. With the 4 smartphones, we evaluate two different network topologies; chain and star topology. We set the device node names with the letters A-B-C-D and we artificially create the two topologies in a lab setting by programmatically removing the corresponding nodes from the available peers during peer discovery. In the experiment scenario, a node initiates a request encapsulating a sensing task, and the time until the last node has received the request is measured. Similarly, when all nodes have executed the task, we initiate

Model	**Specifications**
Samsung Galaxy S7	Android 8.0; Octa-core 2.3 GHz Mongoose; Wi-Fi 802.11 a/b/g/n/ac; 4GB RAM
Motorola G7 Power	Android 10.0; Octa-core 1.8 GHz Kryo 250 Gold; Wi-Fi 802.11 b/g/n; 3GB RAM
Motorola G5 Plus	Android 8.1.0; Octa-core 2.0 GHz Cortex-A53; Wi-Fi 802.11 a/b/g/n/; 3GB RAM
Motorola G4	Android 6.0; Octa-core 1.5 GHz Cortex-A53); Wi-Fi 802.11 a/b/g/n; 3GB RAM

Table 7.1: Specifications of smartphones used in the experiment.

a collect message while measuring the time until all data has arrived. For both topologies, we run the experiment 5 times for each of the nodes, which makes a total of 20 full request-collect cycles for each topology.

Before presenting the results, we go over some specific challenges in fine-tuning the various parameters that affect the experiment. First, since the WiFi Direct API for Android is still relatively new, hardware and especially software variations play a major role on how each device behaves. For instance, the older versions of Android OS, demand a manual input from the user to consent the first time they establish a P2P connection with an unknown device. Android 10, on the other hand, requires an initial manual configuration of the WiFi advanced settings in order for the device to start advertising its group. This is a hindrance for the effectiveness of our application since the objective is for the whole functionality to be passive and not depend on user action. Another major issue is the case of connection request jams, i.e. when two nodes try to connect with a same third one. The smartphones' behaviour in that case is generally unstable, with different phones having varying recovery times to the stable state, and at the extreme case with a WiFi state reset being necessary. To that end, we have introduced a 2-second delay in the advancing of the message queue, and a 60 second retry delay in the

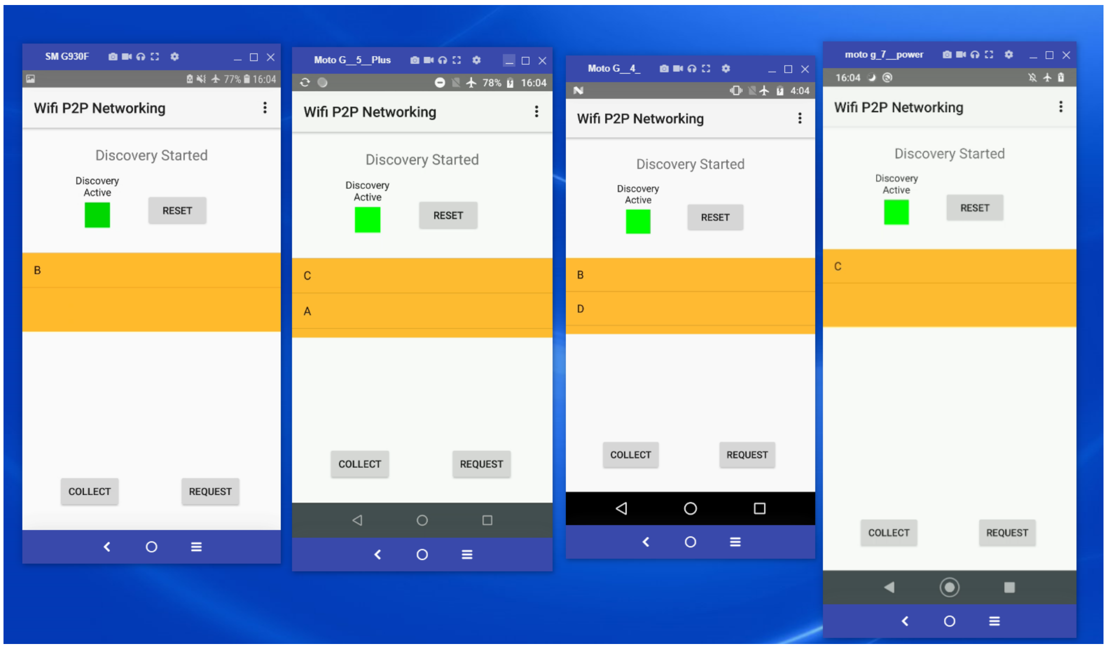


Figure 7.8: The 4 phones set in chain topology. The physical devices are mirrored on a computer monitor using the Vysor visualization software.

case of a failed message sent. These values where chosen empirically, and further research is necessary to conclude to optimal values, but the scope of this work is focused rather on delivering a proof-of-concept implementation, while providing some indicative results reflecting the order of time needed to achieve the desired behavior (seconds, minutes, etc).

In Figure 7.9 we present a graph displaying the average times of completion of REQUEST and COLLECT cycles for the two topologies, after 20 repetitions of the experiment. The 20 repetitions refer to successful cycles, i.e without a manual intervention, which for request cycles required ~ 21 attempts (21 for chain and 20 for star) and ~ 24 for collect (22 chain, 25 star). We notice from the graph that the average time for the two topologies does not deviate much; this is an encouraging result as it hints that the communication protocol does not depend too much on the topology, although more experiments are required. Furthermore, we see that collect cycles require almost 5 times more time than requests, which is expected as they include a lot more message transactions. In particular, for chain topology a request cycle comprises of 3 message transactions while a collect cycle of a total of

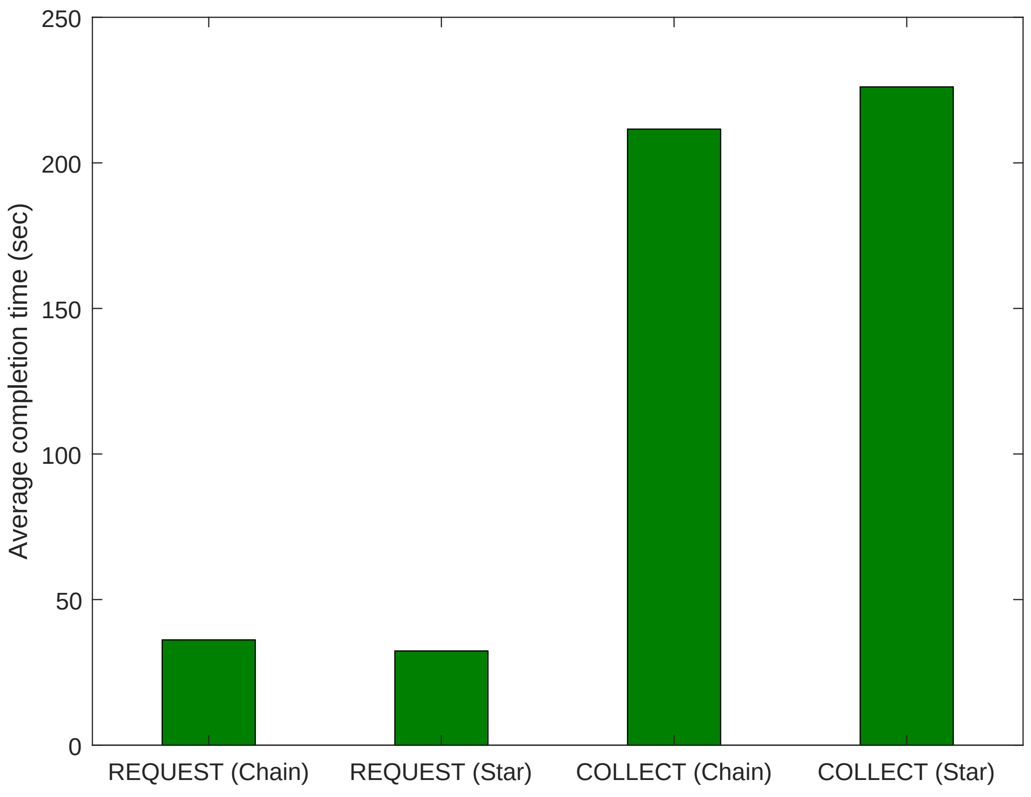


Figure 7.9: Average time of REQUEST and COLLECT successful cycles for chain and star topology.

7, and for star topology the corresponding values are 3 for request and 6 or 8 for collect depending on the originator node. Interestingly, we see that star topology slightly outperforms the chain topology in request completion average time, which is somehow counter-intuitive; this could be explained though if we consider that a central node manages better a sequence of messages to be relayed hence minimizing the transmission errors. This is hinted from the range of measured times in both cases, which was larger for chain topology ([17, 62] while for star [20, 51]). Finally, in collect cycles the star topology is slightly slower, which is expected given the higher probability of jams in connection requests. The range for collect cycles was [92, 321] for chain topology and [88, 348] for star.

7.5 Conclusion

In this chapter we focused on the networking aspects of a crowdsourced framework connecting smartphone users in local proximity. The objective was to enable peers to disseminate requests for data/services in the crowdsourced network, and receive back as many data responses. To enable this behavior a custom data collection protocol was proposed, RE-COLLECT, which is based on the reactive protocol for MANETs DSR and the Directed Diffusion protocol used in IoT. Regarding the implementation, the goal was to provide a solution which can be easily adopted, therefore without too many requirements from the user (expensive equipment, advanced technical skills), in order to favor crowd participation which is paramount in crowdsourced systems. To that end, after reviewing the available options, we choose as the networking medium for this work the WiFi Direct standard, because of its high compatibility in the spectrum of market smartphones, as well as its open set of APIs in Android OS. The latter means that no rooting of smartphone devices is required, something that would significantly hinder the ease of use of our proposal. We deal with the main limitation of WiFi direct, the fact that it only allows for 1-hop network formation, using a cascading device-to-device connection as as workaround. In order to evaluate the above proposed solutions, an Android Application was developed compatible with the vast majority of market devices. Via the mobile app, we test the multi-hop communication workaround technique as well as the RE-COLLECT protocol using 4 different smartphones and two different network topologies. We obtain good results indicating topology invariance of the proposed methods, as well as showcasing the functionality of our solution in a proof-of-concept manner.

Part V

Conclusion

8 Conclusion

Decentralized, flexible, architectures are increasingly needed in networked systems, following the tendency to incorporate large magnitudes of nodes, as well as external system components, often in an ad-hoc manner. In this thesis, we conducted research towards efficient system design, architecture and underlying networking mechanisms, in next-generation IoT networked systems.

Routing in Wireless Ad-Hoc Networks

In terms of routing in wireless ad-hoc networks, one of the biggest challenges faced is designing routing protocols that are sufficiently lightweight, good at balancing the load, and at the same time robust enough to fulfill the system communication requirements. Among the works presented in this thesis, we defined two routing algorithms (Chapter 2 and 7), which both tackle the above challenges. The algorithm detailed in chapter 2, uses a heuristic that utilizes only local network knowledge to efficiently forward publish/subscribe messages, with its experimental results revealing optimal parameter values. In Chapter 8 we define a routing protocol for data collection in similar network settings, which relies on controlled flooding and route piggy-backing.

WSN and Smart Spaces

System architecture plays an important role for designing efficient systems in the context of WSN and smart spaces. Abstraction layers are needed to achieve the necessary interoperability and allow external components to be easily integrated in the system. In Chapter 3 and 4 we presented two cases of fully-fledged systems, one enabling location-based automation for office occupants and one providing a user-friendly tool for visualizing and augmenting a WSN testbed's resources. The location-based automation is enabled via a lightweight adaptation of a WiFi RSSI localization algorithm for smartphones, integrated within a WSN testbed controlling office appliances (Chapter 3). In Chapter 4, we presented an interactive platform modelling in 3D a physical testbed space, as well as its resources, in an on-line manner, while allowing the user to augment it with virtual resources. Both systems are designed with a modular architecture, allowing for smaller sub-parts to be gradually integrated, thus allowing for a more flexible development strategy.

Crowdsourced Systems

Crowdsourced systems are systems whose constituent infrastructure is pooled by the general public via crowdsourcing methodologies. With the proliferation of highly equipped mobile devices as well as personal SBCs, the approach of crowdsourced systems in the field of IoT has become more attractive. In Chapter 5, we identified the formal characteristics, architecture, components and requirements of IoT-related crowdsourced systems, exemplifying them in a use-case regarding crowdsourced cloud computing. Based on the same design principles, in chapter 6, we present a framework implementing edge computing as a crowdsourced system. Finally, in chapter 7, we build a collaborative P2P framework interconnecting smartphones in local proximity, allowing for crowdsourcing of tasks from one peer to another. All of the above implementations are inline with the crowdsourced system principles, outlined in Chapter 5, while they also make use of the results and insights acquired in the two previous thesis parts.

Part VI

Appendix

Projects

Th research conducted in this thesis was supported by the following research projects:

IoT Lab is a project funded by the European Commission (7th Framework Program for Research and Technological Development - FP7) which aims at researching the potential of crowdsourcing to extend IoT testbed infrastructure for multidisciplinary experiments with more end-user interactions. *http://www.iotlab.eu/*

The **SwissSenseSynergy** is a project funded by the Swiss National Science Foundation (SNSF) which aims to provide a unifying framework for secure localisation and privacy-preserving location-based services. *http://www.swiss-sense-synergy.ch/*

Code Repositories

Code from the various applications developed in this thesis can be found at the TCS-Lab's Github page (https://github.com/tcslab). In particular, one will find specific repositories containing:

- Source code of the smartphone application allowing for location-based automation, presented in chapter 3:
  https://github.com/tcslab/Syndesi-3.0—Smartphone-App
- Source code of script and APIs created to automatically query sensor data from Syndesi resources, for the creation of the dataset used in modelled testbeds, presented in chapter 4: https://github.com/tcslab/SenGenEngine, https://github.com/tcslab/SenGen-APIs
- Source code of the smartphone application described in chapter 7, which interconnects phones in local proximity using WiFi Direct and provides a task sharing mechanism for crowdsourcing:
  https://github.com/tcslab/WiFi-P2P-Android-App

Bibliography

[1] N. Belapurkar, J. Harbour, S. Shelke, and B. Aksanli, “Building data-aware and energy-efficient smart spaces,” *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4526–4537, 2018.

[2] S. Kumar, D. Cifuentes, S. Gollakota, and D. Katabi, “Bringing cross-layer mimo to today’s wireless lans,” in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, 2013, pp. 387–398.

[3] A. Sys, “Xm1000 sensor mote,” https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html.

[4] Zolertia, “Z1 sensor mote,” http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf.

[5] A. Katmada, A. Satsiou, and I. Kompatsiaris, “Incentive mechanisms for crowdsourcing platforms,” in *International conference on internet science*. Springer, 2016, pp. 3–18.

[6] H.-C. Hsieh, J.-L. Chen, and A. Benslimane, “5g virtualized multi-access edge computing platform for iot applications,” *Journal of Network and Computer Applications*, vol. 115, pp. 94–102, 2018.

[7] Y. Liu, M. J. Lee, and Y. Zheng, “Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398–2410, 2015.

[8] “https://https://www.tensorflow.org/lite/performance/benchmarks,” TensorFlow Lite Performance benchmarks on mobile devices.

[9] Statista, “Internet users statistics,” https://www.statista.com/statistics/617136/digital-population-worldwide/, 2021.

[10] H. Ehsan and Z. Uzmi, “Performance comparison of ad hoc wireless network routing protocols,” in *8th International Multitopic Conference, 2004. Proceedings of INMIC 2004.*, 2004, pp. 457–465.

[11] C. Li, H. Zhang, B. Hao, and J. Li, “A survey on routing protocols for large-scale wireless sensor networks,” *Sensors*, vol. 11, no. 4, pp. 3498–3526, 2011.

[12] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, “Efficient algorithms for multi-path link-state routing,” 1999.

[13] D. Waitzman, C. Partridge, S. Deering *et al.*, “Distance vector multicast routing protocol,” 1988.

[14] S. Das, R. Castaneda, J. Yan, and R. Sengupta, “Comparative performance evaluation of routing protocols for mobile, ad hoc networks,” in *Proceedings 7th International Conference on Computer Communications and Networks (Cat. No.98EX226)*, 1998, pp. 153–161.

[15] J. Lyklema, C. Evertsz, and L. Pietronero, “The laplacian random walk,” *EPL (Europhysics Letters)*, vol. 2, no. 2, p. 77, 1986.

[16] D. Braginsky and D. Estrin, “Rumor routing algorthim for sensor networks,” in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.* ACM, 2002, pp. 22–31.

[17] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed diffusion: a scalable and robust communication paradigm for sensor networks,” in *Proceedings of the 6th annual international conference on Mobile computing and networking.* ACM, 2000, pp. 56–67.

[18] J. Faruque, K. Psounis, and A. Helmy, “Analysis of gradient-based routing protocols in sensor networks,” in *Distributed Computing in Sensor Systems.* Springer, 2005, pp. 258–275.

[19] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in *Middleware 2001.* Springer, 2001, pp. 329–350.

[20] M. Castro, P. Druschel, A.-M. Kermarrec, and A. I. Rowstron, “Scribe: A large-scale and decentralized application-level multicast infrastructure,” *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1489–1499, 2002.

[21] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, and J. Kelner, “Mires: a publish/subscribe middleware for sensor networks,” *Personal and Ubiquitous Computing*, vol. 10, no. 1, pp. 37–44, 2006.

[22] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “Mqtt-s—a publish/subscribe protocol for wireless sensor networks,” in *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on.* IEEE, 2008, pp. 791–798.

[23] Y. Huang and H. Garcia-Molina, “Publish/subscribe tree construction in wireless ad-hoc networks,” in *Mobile Data Management.* Springer, 2003, pp. 122–140.

[24] A. Giridhar and P. Kumar, “Maximizing the functional lifetime of sensor networks,” in *Proceedings of the 4th international symposium on Information processing in sensor networks.* IEEE Press, 2005, p. 2.

[25] A. Jarry, P. Leone, S. Nikoletseas, and J. Rolim, “Optimal data gathering paths and energy-balance mechanisms in wireless networks,” *Ad Hoc Networks*, vol. 9, no. 6, pp. 1036–1048, 2011.

[26] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, “Information-centric networking: seeing the forest for the trees,” in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, 2011, pp. 1–6.

[27] V. Jacobson, M. Mosko, D. Smetters, and J. Garcia-Luna-Aceves, “Content-centric networking,” *Whitepaper, Palo Alto Research Center*, pp. 2–4, 2007.

[28] B. Krishnamachari, D. Estrin, and S. Wicker, “Modelling data-centric routing in wireless sensor networks,” in *IEEE infocom*, vol. 2. Citeseer, 2002, pp. 39–44.

[29] P. Costa, G. P. Picco, and S. Rossetto, “Publish-subscribe on sensor networks: A semi-probabilistic approach,” in *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on.* IEEE, 2005, pp. 10–pp.

[30] R. Sarkar, X. Zhu, and J. Gao, “Double rulings for information brokerage in sensor networks,” *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 6, pp. 1902–1915, 2009.

[31] C. Scheideler, *Universal Strategies for Interconnection Networks.* Springer, 1998, vol. 1390.

[32] P. Leone and C. Muñoz, “Content based routing with directional random walk for failure tolerance and detection in cooperative large scale wireless networks,” in *SAFECOMP 2013 - Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems) of the 32nd International Conference on Computer Safety, Reliability and Security, Toulouse, France, 2013*, 2013. [Online]. Available: http://hal.archives-ouvertes.fr/SAFECOMP2013-ASCOMS/hal-00848043

[33] D. J. Cook and S. K. Das, “How smart are our environments? an updated look at the state of the art,” *Pervasive and mobile computing*, vol. 3, no. 2, pp. 53–73, 2007.

[34] J. Krumm, *Ubiquitous computing fundamentals.* CRC Press, 2018.

[35] Zolertia, “Z1 sensor,” http://zolertia.io/z1/.

[36] Crossbow, “Telosb sensor,” https://www.willow.co.uk/TelosB_Datasheet.pdf.

[37] J. Y. Kim, H.-J. Lee, J.-Y. Son, and J.-H. Park, “Smart home web of objects-based iot management model and methods for home data mining,”

in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS).* IEEE, 2015, pp. 327–331.

[38] D. Y. Fong, "Wireless sensor networks," *Internet of Things and Data Analytics Handbook*, pp. 197–213, 2017.

[39] C. Worlu, A. A. Jamal, and N. A. Mahiddin, "Wireless sensor networks, internet of things, and their challenges," *Wirel. Sens. Netw*, vol. 8, no. 11, 2019.

[40] D. Christin, A. Reinhardt, P. S. Mogre, R. Steinmetz *et al.*, "Wireless sensor networks and the internet of things: selected challenges," *Proceedings of the 8th GI/ITG KuVS Fachgespräch Drahtlose sensornetze*, pp. 31–34, 2009.

[41] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the internet of things: A survey," in *SoftCOM 2011, 19th international conference on software, telecommunications and computer networks.* IEEE, 2011, pp. 1–6.

[42] M. Kocakulak and I. Butun, "An overview of wireless sensor networks towards internet of things," in *2017 IEEE 7th annual computing and communication workshop and conference (CCWC).* IEEE, 2017, pp. 1–6.

[43] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "Iot gateway: Bridgingwireless sensor networks into internet of things," in *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing.* Ieee, 2010, pp. 347–352.

[44] G. M. Mendoza-Silva, J. Torres-Sospedra, and J. Huerta, "A meta-review of indoor positioning systems," *Sensors*, vol. 19, no. 20, p. 4507, 2019.

[45] V. Stanford, "Using pervasive computing to deliver elder care," *IEEE Pervasive computing*, vol. 1, no. 1, pp. 10–13, 2002.

[46] A. Fleury, M. Vacher, and N. Noury, "Svm-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results," *IEEE transactions on information technology in biomedicine*, vol. 14, no. 2, pp. 274–283, 2009.

[47] Y.-L. Zheng, X.-R. Ding, C. C. Y. Poon, B. P. L. Lo, H. Zhang, X.-L. Zhou, G.-Z. Yang, N. Zhao, and Y.-T. Zhang, "Unobtrusive sensing and wearable devices for health informatics," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 5, pp. 1538–1554, 2014.

[48] M. Alwan, P. J. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder, "A smart and passive floor-vibration based fall detector for elderly," in *2006 2nd International Conference on Information Communication Technologies*, vol. 1. IEEE, 2006, pp. 1003–1007.

[49] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 254–263.

[50] C. Charachristos, C. Goumopoulos, and A. Kameas, "Enhancing collaborative learning and management tasks through pervasive computing technologies," *Ambient Computing, Applications, Services and Technologies*, pp. 27–33, 2014.

[51] H. Bargaoui and R. Bdiwi, "Smart classroom: Design of a gateway for ubiquitous classroom," in *2014 International Conference on Web and Open Access to Learning (ICWOAL)*. IEEE, 2014, pp. 1–4.

[52] D.-m. Han and J.-h. Lim, "Smart home energy management system using ieee 802.15.4 and zigbee," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1403–1410, 2010.

[53] J. Byun, B. Jeon, J. Noh, Y. Kim, and S. Park, "An intelligent self-adjusting sensor for smart home services based on zigbee communications," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 3, pp. 794–802, 2012.

[54] H. Y. Wang and J. Y. Wang, "Design and implementation of a smart home based on wsn and amr," in *Applied Mechanics and Materials*, vol. 271. Trans Tech Publ, 2013, pp. 1485–1489.

[55] J. M. Wang and H. B. Wei, "Design of smart home management system based on gsm and zigbee," in *Advanced Materials Research*, vol. 842. Trans Tech Publ, 2014, pp. 703–707.

[56] W. Liu and Y. Yan, "Application of zigbee wireless sensor network in smart home system," *International Journal of Advancements in Computing Technology*, vol. 3, no. 5, 2011.

[57] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE transactions on industrial electronics*, vol. 57, no. 10, pp. 3557–3564, 2010.

[58] X. Cao, J. Chen, Y. Xiao, and Y. Sun, "Building-environment control with wireless sensor and actuator networks: Centralized versus distributed," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3596–3605, 2009.

[59] M. Magno, D. Boyle, D. Brunelli, B. O'Flynn, E. Popovici, and L. Benini, "Extended wireless monitoring through intelligent hybrid energy supply," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 4, pp. 1871–1881, 2013.

[60] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay, "Towards the implementation of iot for environmental condition monitoring in homes," *IEEE sensors journal*, vol. 13, no. 10, pp. 3846–3853, 2013.

[61] S. C. Ergen, "Zigbee/ieee 802.15. 4 summary," *UC Berkeley, September*, vol. 10, no. 17, p. 11, 2004.

[62] J. C. Haartsen, "The bluetooth radio system," *IEEE personal communications*, vol. 7, no. 1, pp. 28–36, 2000.

[63] G. Mulligan, "The 6lowpan architecture," in *Proceedings of the 4th workshop on Embedded networked sensors*, 2007, pp. 78–82.

[64] X. Wang and S. Zhong, "A hierarchical scheme on achieving all-ip communication between wsn and ipv6 networks," *AEU-International Journal of Electronics and Communications*, vol. 67, no. 5, pp. 414–425, 2013.

[65] L. Palma, L. Pernini, A. Belli, S. Valenti, L. Maurizi, and P. Pierleoni, "Ipv6 wsn solution for integration and interoperation between smart home and aal systems," in *2016 IEEE Sensors Applications Symposium (SAS).* IEEE, 2016, pp. 1–5.

[66] A. J. Jabir, S. K. Subramaniam, Z. Z. Ahmad, and N. A. W. A. Hamid, "A cluster-based proxy mobile ipv6 for ip-wsns," *EURASIP Journal on Wireless Communications and networking*, vol. 2012, no. 1, pp. 1–17, 2012.

[67] S. Kumar, N. Lal, and V. K. Chaurasiya, "An energy efficient ipv6 packet delivery scheme for industrial iot over g. 9959 protocol based wireless sensor network (wsn)," *Computer Networks*, vol. 154, pp. 79–87, 2019.

[68] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "Tinyos: An operating system for sensor networks," in *Ambient intelligence.* Springer, 2005, pp. 115–148.

[69] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on.* IEEE, 2004, pp. 455–462.

[70] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[71] N.-T. Dinh and Y.-H. Kim, "Restful architecture of wireless sensor network for building management system," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 6, no. 1, pp. 46–63, 2012.

[72] O. Evangelatos, K. Samarasinghe, and J. Rolim, "Syndesi: A framework for creating personalized smart environments using wireless sensor networks," in *2013 IEEE international conference on distributed computing in sensor systems.* IEEE, 2013, pp. 325–330.

[73] S. Yiu, M. Dashti, H. Claussen, and F. Perez-Cruz, "Wireless rssi fingerprinting localization," *Signal Processing*, vol. 131, pp. 235–244, 2017.

[74] K. Whitehouse, C. Karlof, and D. Culler, “A practical evaluation of radio signal strength for ranging-based localization,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 1, pp. 41–52, 2007.

[75] K. Muthukrishnan, G. Koprinkov, N. Meratnia, and M. Lijding, “Using time-of-flight for wlan localization: feasibility study,” *Centre for Telematics and Information Technology (CTIT), University of Twente, Technical Report, no. TR-CTIT-06–28*, 2006.

[76] Y.-T. Chan, W.-Y. Tsui, H.-C. So, and P.-c. Ching, “Time-of-arrival based localization under nlos conditions,” *IEEE Transactions on Vehicular Technology*, vol. 55, no. 1, pp. 17–24, 2006.

[77] A. Awad, T. Frunzke, and F. Dressler, “Adaptive distance estimation and localization in wsn using rssi measures,” in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*. IEEE, 2007, pp. 471–478.

[78] F. Potortì, S. Park, A. R. Jimenez Ruiz, P. Barsocchi, M. Girolami, A. Crivello, S. Y. Lee, J. H. Lim, J. Torres-Sospedra, F. Seco *et al.*, “Comparing the performance of indoor localization systems through the evaal framework,” *Sensors*, vol. 17, no. 10, p. 2327, 2017.

[79] G. Deak, K. Curran, and J. Condell, “A survey of active and passive indoor localisation systems,” *Computer Communications*, vol. 35, no. 16, pp. 1939–1954, 2012.

[80] X. Zhao, Z. Xiao, A. Markham, N. Trigoni, and Y. Ren, “Does btle measure up against wifi? a comparison of indoor location performance,” in *European Wireless 2014; 20th European Wireless Conference*. VDE, 2014, pp. 1–6.

[81] S. Beauregard and H. Haas, “Pedestrian dead reckoning: A basis for personal positioning,” in *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*, 2006, pp. 27–35.

[82] R. Zhang, W. Xia, Z. Jia, and L. Shen, “The indoor localization method based on the integration of rssi and inertial sensor,” in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. IEEE, 2014, pp. 332–336.

[83] W. W.-L. Li, R. A. Iltis, and M. Z. Win, “A smartphone localization algorithm using rssi and inertial sensor measurement fusion,” in *2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2013, pp. 3335–3340.

[84] V. Malyavej, W. Kumkeaw, and M. Aorpimai, “Indoor robot localization by rssi/imu sensor fusion,” in *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. IEEE, 2013, pp. 1–6.

[85] X. Li, J. Wang, and C. Liu, “A bluetooth/pdr integration algorithm for an indoor positioning system,” *Sensors*, vol. 15, no. 10, pp. 24 862–24 885, 2015.

[86] Z. Wu, E. Jedari, R. Muscedere, and R. Rashidzadeh, “Improved particle filter based on wlan rssi fingerprinting and smart sensors for indoor localization,” *Computer Communications*, vol. 83, pp. 64–71, 2016.

[87] J. L. Carrera, Z. Li, Z. Zhao, T. Braun, and A. Neto, “A real-time indoor tracking system in smartphones,” in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2016, pp. 292–301.

[88] Z. Li, T. Braun, and D. C. Dimitrova, “A passive wifi source localization system based on fine-grained power-based trilateration,” in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2015, pp. 1–9.

[89] L. E. Peterson, “K-nearest neighbor,” *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[90] S. Suthaharan, “Support vector machine,” in *Machine learning models and algorithms for big data classification*. Springer, 2016, pp. 207–235.

[91] A. OS, “Releases statistics,” https://developer.android.com/about/dashboards/index.html.

[92] J. Bucanek, “Model-view-controller pattern,” *Learn Objective-C for Java Developers*, pp. 353–402, 2009.

[93] D. Jackson and E. Kang, “Separation of concerns for dependable software design,” in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 173–176.

[94] G. Bradski and A. Kaehler, “Opencv,” *Dr. Dobb’s journal of software tools*, vol. 3, 2000.

[95] “Fed4fire project.” [Online]. Available: http://www.fed4fire.eu/

[96] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, “Fit iot-lab: A large scale open experimental iot testbed,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 459–464.

[97] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, “Wisebed: an open large-scale wireless sensor network testbed,” in *International Conference on Sensor Applications, Experimentation and Logistics*. Springer, 2009, pp. 68–87.

[98] X. Ju, H. Zhang, and D. Sakamuri, “Neteye: a user-centered wireless sensor network testbed for high-fidelity, robust experimentation,” *International Journal of Communication Systems*, vol. 25, no. 9, pp. 1213–1229, 2012.

[99] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, “Smartsantander: Iot experimentation over a smart city testbed,” *Computer Networks*, vol. 61, pp. 217–238, 2014.

[100] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, “Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems,” in *Proceedings of the 12th international conference on Information processing in sensor networks*, 2013, pp. 153–166.

[101] A.-S. Tonneau, N. Mitton, and J. Vandaele, “A survey on (mobile) wireless sensor network experimentation testbeds,” in *2014 IEEE International Conference on Distributed Computing in Sensor Systems*, 2014, pp. 263–268.

[102] A. Boukerche, S. K. Das, and A. Fabbri, “Swimnet: A scalable parallel simulation testbed for wireless and mobile networks,” *Wireless Networks*, vol. 7, no. 5, pp. 467–486, 2001.

[103] S. Cavalieri, M. Macchi, and P. Valckenaers, “Benchmarking the performance of manufacturing control systems: design principles for a web-based simulated testbed,” *J. Intelligent Manufacturing*, vol. 14, no. 1, pp. 43–58, 2003. [Online]. Available: http://dx.doi.org/10.1023/A%3A1022287212706

[104] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. A. Syed, S. Sharma, and T.-c. Chiueh, “Mint-m: an autonomous mobile wireless experimentation platform,” in *Proceedings of the 4th international conference on Mobile Systems, Applications and Services*, 2006, pp. 124–137.

[105] L. Baron, R. Klacza, M. Y. Rahman, C. Scognamiglio, N. Kurose, T. Friedman, and S. Fdida, “Onelab tutorial: A single portal to heterogeneous testbeds,” *EAI Endorsed Trans. Ubiquitous Environments*, vol. 2, no. 6, p. e4, 2015. [Online]. Available: http://dx.doi.org/10.4108/icst.tridentcom.2015.259876

[106] J. Surowiecki, M. P. Silverman *et al.*, “The wisdom of crowds,” *American Journal of Physics*, vol. 75, no. 2, pp. 190–192, 2007.

[107] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, “gmission: A general spatial crowdsourcing platform,” *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1629–1632, 2014.

[108] L. F. G. Sarmenta, “Volunteer computing,” Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[109] C. Ernst, A. Mladenow, and C. Strauss, “Collaboration and crowdsourcing in emergency management,” *International Journal of Pervasive Computing and Communications*, 2017.

[110] G. D. Saxton, O. Oh, and R. Kishore, “Rules of crowdsourcing: Models, issues, and systems of control,” *Information Systems Management*, vol. 30, no. 1, pp. 2–20, 2013.

[111] F. Galton, “Vox populi (the wisdom of crowds),” *Nature*, vol. 75, no. 7, pp. 450–451, 1907.

[112] E. Ostrom, “The difference: How the power of diversity creates better groups, firms, schools, and societies. by scott e. page. princeton: Princeton university press, 2007. 448p. 19.95 paper.” *Perspectives on Politics*, vol. 6, no. 4, pp. 828–829, 2008.

[113] Float (last accessed on 14/11/2017). [Online]. Available: http://civicus.org/thedatashift/wp-content/uploads/2015/07/Float-Beijing-case-study.pdf

[114] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, “Seti@home: an experiment in public-resource computing,” *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[115] G. I. of Medical Research, “Dreamlab,” https://www.garvan.org.au/support-us/dreamlab, 2021.

[116] I. Laponogov, G. Gonzalez, M. Shepherd, A. Qureshi, D. Veselkov, G. Charkoftaki, V. Vasiliou, J. Youssef, R. Mirnezami, M. Bronstein *et al.*, “Network machine learning maps phytochemically rich “hyperfoods” to fight covid-19,” *Human genomics*, vol. 15, no. 1, pp. 1–11, 2021.

[117] A. Spark, “Apache spark: Lightning-fast cluster computing,” *URL http://spark. apache. org*, 2016.

[118] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, “Apache spark: A unified engine for big data processing,” *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016.

[119] Statista, “Worldwide connected device statistics,” https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/, 2021.

[120] S. Yi, Z. Hao, Z. Qin, and Q. Li, “Fog computing: Platform and applications,” in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Nov 2015.

[121] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: A platform for internet of things and analytics,” in *Big data and internet of things: A roadmap for smart environments.* Springer, 2014, pp. 169–186.

[122] M. Satyanarayanan, “The emergence of edge computing,” *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[123] ITU-T, “Requirements of the imt-2020 network,” International Telecommunication Union, Geneva, Recommendation Y.3101, Nov. 2018.

[124] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing—a key technology towards 5g,” *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[125] Gartner, https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends, 2018.

[126] M. Hosseini, C. M. Angelopoulos, W. K. Chai, and S. B. Kundig, “Crowdcloud: a crowdsourced system for cloud infrastructure,” *Cluster Computing*, pp. 1–16, 2018.

[127] The Directorate-General for Research and Innovation, European Commission, “Vision and trends of social innovation for europe,” 2017. [Online]. Available: https://publications.europa.eu/en/publication-detail/-/publication/a97a2fbd-b7da-11e7-837e-01aa75ed71a1#r

[128] CAPSSI-EU, “Network infrastructure as commons,” https://capssi.eu/network-infrastructure-as-commons/, 2018.

[129] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. New York, NY, USA: ACM, 2015, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/2757384.2757397

[130] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, Nov 2016, pp. 20–26.

[131] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[132] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[133] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, 2012.

[134] I. S. 1934, "Standard for adoption of openfog reference architecture for fog computing," *IEEE Standards Association*, 2018.

[135] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[136] IBM, "Smarter wireless networks: Add intelligence to the mobile network edge," https://www-935.ibm.com/services/multimedia/Smarter_wireless_networks.pdf, 2013.

[137] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[138] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[139] V. Bahl, "emergence of micro datacenter (cloudlets/edges) for mobile computing," *Online: https://www. microsoft. com/en-us/research/wp-content/uploads/2016/11/Micro-Data-Centers-mDCs-for-Mobile-Computing-1. pdf. Accessed*, vol. 12, 2017.

[140] K. Ha and M. Satyanarayanan, "Openstack++ for cloudlet deployment," *School of Computer Science Carnegie Mellon University Pittsburgh*, 2015.

[141] M. Marjanovic, A. Antonic, and I. P. Zarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, 2018.

[142] P. Bellavista, S. Chessa, L. Foschini, L. Gioia, and M. Girolami, "Human-Enabled Edge Computing: Exploiting the Crowd as a Dynamic Extension of Mobile Edge Computing," *IEEE Communications Magazine*, 2018.

[143] P. Antoniadis, J. Ott, and A. Passarella, "Do it yourself networking: an interdisciplinary approach (dagstuhl seminar 14042)," in *Dagstuhl reports*, vol. 4, no. 1. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[144] "https://github.com/Wei1234c/Broccoli," Broccoli stack.

[145] I. Wells, "High speed packet processing using a distributed hash table," Aug. 8 2017, uS Patent 9,729,444.

[146] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, 2014.

[147] B. I. Ismail, E. M. Goortani, M. B. Ab Karim, W. M. Tat, S. Setapa, J. Y. Luke, and O. H. Hoe, "Evaluation of docker as edge computing platform," in *Open Systems (ICOS), 2015 IEEE Confernece on.* IEEE, 2015, pp. 130–135.

[148] A. Gotsman, H. Yang, C. Ferreira, M. Najafzadeh, and M. Shapiro, "'cause i'm strong enough: Reasoning about consistency choices in distributed systems,"

in *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2016, pp. 371–384.

[149] D. Bermbach and J. Kuhlenkamp, "Consistency in distributed storage systems," in *International Conference on Networked Systems*. Springer, 2013, pp. 175–189.

[150] "http://edgent.apache.org/," Apache Edgent.

[151] S. Gaglio, G. L. Re, G. Martorella, and D. Peri, "Dc4cd: A platform for distributed computing on constrained devices," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 1, Dec. 2017.

[152] "Facebook data-breach," https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html/.

[153] "Google+ data-breach," https://www.sciencealert.com/google-kept-a-bug-secret-for-months-that-risked-google-users-privacy.

[154] M. Langheinrich, "Privacy by design—principles of privacy-aware ubiquitous systems," in *International conference on Ubiquitous Computing*. Springer, 2001, pp. 273–291.

[155] C. M. Angelopoulos, S. E. Nikoletseas, T. P. Raptis, and J. D. P. Rolim, "Characteristic utilities, join policies and efficient incentives in mobile crowdsensing systems," in *2014 IFIP Wireless Days, WD 2014, Rio de Janeiro, Brazil, November 12-14, 2014*, 2014, pp. 1–6.

[156] Akamai, https://www.akamai.com/about/, 2018.

[157] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[158] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[159] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[160] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[161] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and$<$ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[162] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[163] "https://www.tensorflow.org/lite," TensorFlow Lite.

[164] "https://gs.statcounter.com/os-market-share/mobile/worldwide," Mobile OS market share statistics, StatCounter.

[165] "https://developers.google.com/maps/documentation/android-sdk/utility/heatmap," Google Maps Heatmap Utility.

[166] P. Antoniadis and I. Apostol, "The right (s) to the hybrid city and the role of diy networking," *The Journal of Community Informatics*, vol. 10, no. 3, 2014.

[167] A. Gulati, M. Sharma, N. Sharma, and S. Bhatia, "Ad-hoc network and their applications."

[168] Statista, "Smartphone users statistics," https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, 2021.

[169] M. Baert, J. Rossey, A. Shahid, and J. Hoebeke, "The bluetooth mesh standard: An overview and experimental evaluation," *Sensors*, vol. 18, no. 8, p. 2409, 2018.

[170] M. R. Ghori, T.-C. Wan, and G. C. Sodhy, "Bluetooth low energy mesh networks: Survey of communication and security protocols," *Sensors*, vol. 20, no. 12, p. 3590, 2020.

[171] J. Thomas, J. Robble, and N. Modly, "Off grid communications with android meshing the mobile world," in *2012 IEEE Conference on Technologies for Homeland Security (HST)*. IEEE, 2012, pp. 401–405.

[172] J. Jubin and J. D. Tornow, "The darpa packet radio network protocols," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 21–32, 1987.

[173] S. Mohseni, R. Hassan, A. Patel, and R. Razali, "Comparative review study of reactive and proactive routing protocols in manets," in *4th IEEE International Conference on Digital Ecosystems and Technologies*, 2010, pp. 304–309.

[174] C. Funai, C. Tapparello, and W. Heinzelman, "Enabling multi-hop ad hoc networks through wifi direct multi-group networking," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 491–497.

[175] Y. Duan, C. Borgiattino, C. Casetti, C. F. Chiasserini, P. Giaccone, M. Ricca, F. Malabocchia, and M. Turolla, "Wi-fi direct multi-group data dissemination for public safety," in *WTC 2014; World Telecommunications Congress 2014*. VDE, 2014, pp. 1–6.

[176] A. Bhojan and G. W. Tan, "Mumble: Framework for seamless message transfer on smartphones," in *Proceedings of the 1st International Workshop on Experiences with the Design and Implementation of Smart Objects*, 2015, pp. 43–48.

[177] K. Liu, W. Shen, B. Yin, X. Cao, L. X. Cai, and Y. Cheng, "Development of mobile ad-hoc networks over wi-fi direct with off-the-shelf android phones," in *2016 IEEE international conference on communications (ICC)*. IEEE, 2016, pp. 1–6.