AI based localisation methods on low power devices

Honnalli, Siddhant R.

Award date:
2021

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Electro-Optical Communications Research Group

AI based localisation methods on low power devices

Master Thesis

Siddhant Rajashekar Honnalli
s.r.honnalli@student.tue.nl

Supervisors:
dr. Georgios Exarchakos g.exarchakos@tue.nl
dr. Tanir Ozcelebi t.ozcelebi@tue.nl
prof.dr.ir. Nirvana Meratnia n.meratnia@tue.nl

Eindhoven, March 2021

Abstract

In wireless communications, the propagation medium poses many challenges which forces to improve the transceiver architecture. The medium has always been observed to exhibit random behaviour, which degrades the quality of the signal at receiver end due to the uncontrollable interactions of the transmitted radio waves with the surrounding objects. The advent of reconfigurable intelligent surface (RIS) is changing this observation. Empowering RISs with the knowledge of location of target receiver improves signal strength. The surfaces consume very little energy and assuming RISs can be powered from harvested energy in future, requires energy efficient localisation.

This master thesis proposes an implementation of fingerprinting based localisation methods that perform the inference of target device location from simple radio strength measurements. The system consists of wirelessly connected nodes that entirely work on harvested energy and does not require external resources for computation when making predictions. Experiments are performed with two localisation methods and a throughput of 29 predictions per minute has been achieved. The average error of location predicted by the systems is less than one meter.

Preface

The last 9 months was an amazing experience. The project was really challenging and intriguing at the same time. I would like to take this opportunity to thank my supervisor, Georgios Exarchakos for extending his trust and patience towards me. Without his guidance and constant feedback, this master thesis would not have been possible. I would also like to thank assistant professor, Majid Nabi Najafabadi from TU/e, whose course on Networked Embedded Systems introduced me to the amazing world of Wireless Network Systems. I would like to mention student psychologist, Martje van der Horst who helped to keep my mental balance in good shape during the execution of the thesis.

I would like to thank my parents and my sister who are my constant pillars of support at all times and their encouragement to pursue my Master studies in Europe.

I would also like to thank EIT Digital Master School providing me a great opportunity to be a part of two highly reputed European Universities(Technische Universitat Berlin, Germany and Technical University of Eindhoven, Netherlands).

Contents

Contents . . . **vii**

List of Figures . . . **ix**

Listings . . . **xi**

1 Introduction . . . **1**
- 1.1 Problem Statement . . . 3
- 1.2 Assumptions . . . 3
- 1.3 Thesis Structure . . . 4

2 Background . . . **5**
- 2.1 Neural Networks . . . 5
- 2.2 Network communication models . . . 7
  - 2.2.1 OSI model . . . 7
- 2.3 Overview of IEEE 802.15.4e . . . 8
  - 2.3.1 Network Devices . . . 9
  - 2.3.2 Network Topologies . . . 9
  - 2.3.3 Physical Layer . . . 11
  - 2.3.4 MAC Layer . . . 11
- 2.4 Overview of Contiki OS and Contiki-NG . . . 12
  - 2.4.1 Features of Contiki-NG . . . 12
  - 2.4.2 Contiki-NG architecture . . . 13
  - 2.4.3 Primitives of Contiki-NG . . . 13

3 Literature study . . . **15**
- 3.1 AI on low power devices . . . 15
- 3.2 Indoor Localisation . . . 16
- 3.3 Summary . . . 18

4 Sytem Design . . . **19**
- 4.1 Overview . . . 20
  - 4.1.1 Fingerprinting phase . . . 20
  - 4.1.2 Training phase . . . 21

4.1.3 Estimation phase 22
4.2 Design adaptation to Low power device 26

5 Implementation **28**
5.1 Hardware Platfrom 28
5.1.1 Zolertia Firefly 28
5.2 Test bed 29
5.3 Contiki-NG OS 31
5.4 Memory Constraint 33
5.5 Custom Network Stack 34
5.5.1 Data Structures and Functions 34
5.6 Neural Network Optimisation 36
5.7 Experimental set up 36
5.7.1 Gravel net 36
5.7.2 Fingerprinting phase 36
5.7.3 Training Phase 37
5.7.4 Estimation phase 37
5.8 Evaluation metrics 40
5.8.1 Positioning accuracy 40
5.8.2 Rate of Predictions 40
5.8.3 Packet Reception Ratio(PRR) 40

6 Results **41**
6.0.1 Localisation 41
6.0.2 Prediction Rate 42
6.0.3 Packet reception ratio 43

7 Conclusions and future work **45**
7.1 Conclusions 45
7.2 Future Work 45

Bibliography **47**

List of Figures

2.1 Analogy of biological neuron, artificial neuron and sensor node[30] 6
2.2 OSI model[7] 7
2.3 IEEE 802.15.4e/Zigbee network stack[13] 8
2.4 Star Topology[8] 10
2.5 Peer-to-Peer Topology[8] 10
2.6 Tree Topology[8] 11
2.7 Scheduling in Contiki[25] 14

3.1 Architecture of Big-Little feed-forward neural network[15] 16

4.1 AI based localisation 19
4.2 Fingerprinting methodology 22
4.3 Neural network architecture 22
4.4 Connections in Central method with 4 nodes 23
4.5 Central localisation method with 4 nodes 24
4.6 Connections in Distributed method with 4 nodes 25
4.7 Distributed localisation method with 4 nodes 26

5.1 Neural node architecture [14] 28
5.2 CC2538 architecture[21] 29
5.3 Lab floor layout 30
5.4 Deployment of nodes on the ceiling 30
5.5 Network protocol stack of Contiki-NG [35] 31
5.6 TSCH timeslot and example slotframe[17] 32
5.7 Network stack with custom network layer 33
5.8 Localisation error for central and distributed method 37

6.1 Mean localisation error for central method 41
6.2 Mean localisation error for distributed method 42
6.3 Rate of predictions with time with 1packet/s of target user 43
6.4 Packet Reception Ratio in central method 44
6.5 Packet Reception Ratio in distributed method 44

Listings

5.1 Structure of custom network driver in Contiki-NG 34
5.2 Structure of sniffer, input and send functions of custom network layer in Contiki-NG 34

Chapter 1

Introduction

The propagation medium for wireless communications has been passive. Electromagnetic (EM) waves generated by antennas and electromagnetically active elements are subject to attenuation due the nature of the medium and surrounding environments. Especially in an indoor environment, the EM waves deteriorate as they propagate due to different local effects such as multiple reflection, scattering, diffraction, interference and fading [12]. The deterioration is mainly caused because of the material of walls and geometry of building structure. The negative effects result in lower quality of data transmission. These effects increase with more number of networked devices in the building, a trend which is growing. It has been projected that by the year 2022, the number of connections and networked devices will reach up to 28.5 billions, and 12.3 billions of them will consist of mobile-ready devices and connections [19]. Technologies such as 5th Generation (5G) and 6th Generation (6G) mobile communications bring in new use cases and new applications. The new networking trends will come with newer communication engineering problems and challenges, necessitating new communication paradigms, especially at the physical layer. This has led to new research interests to control the wave propagation and make the propagation medium active. One such interest has been the reconfigurable intelligent surfaces (RISs) [10] that actively control the propagation environment to enhance the signal quality at the receiver.

A RIS can be considered as a flat surface that is able to modify the radio waves impinging upon it in ways that a naturally occurring materials are unable to do [10]. RISs can be deployed in many scenarios in indoor and outdoor environments to improve the performance of wireless communication systems. In indoor scenario, covering physical objects or embedding walls with RISs can make the radio environment into a smart space. The surfaces typically operate as reflectors and are made of individually tunable unit elements whose phase response can be individually configured and optimized for focusing, beam steering, and other similar functions. In order to configure RISs to strengthen signal focusing in indoor environments, a variety of ML algorithms have been proposed such as K-nearest neighbors, support vector machines, Bayesian learning and deep learning [20]. To improve the transmitted signal focusing to intended target receiver positions in indoor communication environments as mentioned in [20], it is necessary to know the location of the target receiver. Assuming these surfaces are powered from harvested energy in future, the need of a energy efficient indoor localisation system inherent to the wireless system becomes clear. The principle behind such a wireless localisation system relies on the assumption that multiple randomly dispersed wireless nodes that receive signals could detect the location of the source of that signal if they collaborated to mitigate shadowing and fading problems coming from uncontrolled multi-path propagation and attenuation. This collaboration between multiple wireless nodes have to implement the inherent non-linear function of localisation.

Indoor localisation has been extensively investigated over the last few decades, mainly in industrial settings and for wireless sensor networks and robotics. The popular localisation technology used is Global Position System (GPS) signals or base stations (BSs) in the existing cellular networks. However, these positioning methods are accurate for outdoors but not feasible for indoors, due to the influence of obstacles, limited space size and interference of noise. Therefore, a large number of technologies, such as Bluetooth, radio frequency identification (RFID), wireless local area network (WLAN or Wi-Fi), magnetic field variations, ultrasound, Zigbee, bluetooth low energy (BLE) and light-emitting diode (LED) light bulbs, have been developed to create high-accuracy indoor positioning systems (IPS). Zigbee has a greater range than BLE as it can transmit farther by using a mesh network of relay nodes to reach a destination. Zigbee is commonly used for localisation due to its low power requirements [33]. More importantly, Zigbee is based on IEEE 802.15.4 standard which is built for low power devices.

The received signal strength indicator (RSSI) based indoor localisation is a widely chosen approach as it is easy to implement, cost efficient and can be used with number of technologies. It is basically the measure of power of RF signal received by the device from the transmitter [47]. The IPS algorithm that uses RSSI based indoor localization can be classified into two main types: log-distance propagation model (PM) algorithms based on the signal and fingerprinting indoor localization based on the data collected. IPS based on signal propagation is divided into lateration and angulation. The main idea of lateration estimation is to calculate the distance between the target device and access point or node using geometry and signal measurement information, such as the time of arrival (TOA), time difference of arrival (TDOA), and angle of arrival (AOA), of the incoming signals from access points (APs). In general, propagation signals suffer from non-line-of-sight (NLOS) multipath signals due to the presence of walls and furniture and the movement of people. In addition, the signal accuracy decreases if one or more node coordinates are not accurately calculated. Thus, fingerprinting-based localisation systems have been proposed as an alternative technology. The main advantage of the fingerprinting based localisation technique is that it can provide accurate location estimation in the challenging multipath environments and thus outperforms some common localisation techniques based on lateration and angulation [46]. Fingerprinting based localisation system can be executed on high power servers. However, this would create a single point of failure and/or the end user would not want to install additional hardware connected to mains power. These type of systems are not energy efficient. Another approach is to realise energy efficient localisation on resource-limited low power wireless devices. This approach is more preferable for users as privacy issues are typically associated with the server-based techniques [24]. According to [37], energy harvested from RF waves can power resource constraint wireless devices. The main characteristic of such devices is low power, which usually means limited computing power and small memory size ($\leq 20KB$). On the software side, to reduce the memory footprint, applications usually run on bare metal or use a very lightweight OS with a minimal set of third-party libraries. In [40], the authors show that optimising deep learning models for low power devices, high performance with limited computing power can be achieved. Morever, in [36], the authors estimate that artificial neural network (ANN) used for localisation uses very less amount of memory that could be deployed on low power devices.

This thesis proposes fingerprinting based localisation methods that consists of wirelessly connected nodes. These nodes collaborate with each other to perform localisation based on the RSSI of data transmissions that orginate from the target device. Specifically, we implement two localisation methods: central method and distributed method. In the former method, a single node performs localisation and other nodes in the network collaborate by providing the RSSI inputs to the localising node. In the latter method, all nodes collaborate and each node performs localisation based on data received from all the other nodes.

1.1 Problem Statement

In this thesis, the main goal is to design and implement AI based localisation methods that use fingerprinting approach and optimised ANN models deployed on an IEEE 802.15.4 based network of low power devices that use harvested energy.
Following are the goals of this thesis:

- Design and implement an ANN system for low power devices.
- Deploy the ANN on the chosen operating system (Contiki-NG).
- Perform localisation based on RSSI of target device on a IEEE 802.15.4 network using two methods: central localisation method and distributed localisation method. Evaluate the performance of the methods.

1.2 Assumptions

In order to simplify the problem, this thesis is based on the following assumptions:

- The mechanism to harvest and store energy is already implemented that can power the devices during their lifetime.
- Each low power device or node has a RSS sensor and IEEE 802.15.4 radio for communication. In an indoor environment, the wireless nodes are already randomly deployed on the ceiling.

- The target device transmits periodic broadcast to the wireless nodes.

1.3 Thesis Structure

The structure of the thesis report is as follows. Chapter 2 provides essential background of neural networks, network stacks, and in more detail about IEEE 802.15.4. It also explains in detail about the design and architecture of Contiki-NG. Chapter 3 studies the current relevant research concerning distributed artificial intelligence and indoor localisation. Chapter 4 explains the system design for localisation and ANN optimisation. Chapter 5 provides details about implementation using Contiki-NG with custom developed network stack, experiment setup and evaluation metrics. Chapter 6 presents the results obtained according the metrics and discussion of results. Finally, chapter 7 states the conclusion and proposes a couple of possible topics that can taken up for the future.

Chapter 2

Background

This chapter provides background information of neural networks in wireless sensor networks, wireless technology and Internet of things (IoT) operating systems, specifically Contiki and Contiki-NG. First, an analogy of biological neuron, artificial neuron and sensor node is explained. Second, the theoretical model of network communication is explained. Third, technical details about IEEE 802.15.4e is provided. Finally, an overview of the operating system is provided. All of the above is necessary for understanding the remaining of the report.

2.1 Neural Networks

Many of the characteristics of Artificial Neural Networks (ANN) are desirable for WSNs. ANN show characteristics such as adaptivity, inherent contextual information processing, distributed representation and processing, massive parallelism, learning and generalization ability, fault tolerance and low computation [30].

In classical computer networking, each computer can be viewed as an computing entity which processes and stores data. The data is forwarded to other computers which are connected to the network. Similarly, WSNs could be viewed as a set of tiny processors which are deployed over a region of interest. The computational capabilities of these processors are low. The associated sensor nodes maintain communication links to each other and sense the surrounding environment. In the region of WSN, if an event occurs such as change of temperature and atmospheric pressure, local sensors start to gather data and propagate their information forward to a central location [30]

The human brain is primarily composed of neurons, glial cells, neural stem cells, and blood vessels. The neuron is composed of dendrites, an axon and a cell body called soma. Each neuron can form a link to another neuron via the synapse which is a junction of a dendrite and an axon. Within the synapses, postsynaptic potentials are generated which are received via dendrites and chemically transformed within the soma. The axon transports the action potential sent out by the soma to the next synapse[23]. Fig:2.1 shows the transfer the perception of a biological neuron via an artificial neuron to the sensor node.

In artificial neurons is easy, the incoming signals are weighted, as it is done in synapses, and further processed. The functionality h(x) is basically a weighted sum over all inputs. Here the output corresponds to the axon. The sensor converts the physical world to an electrical signal which is filtered or preprocessed corresponding to weighting/synapse. The subsequent processing within the processor corresponds to the chemical processing accomplished by the soma or applying the particular functionality h(x), respectively. Finally, the sensor node sends out the modified sensor reading via the radio link. This strong analogy shows that the sensor node itself can be seen as a biological and artificial neuron, respectively. Hence, we can easily extend our horizon and see some WSNs as largescale ANNs whereas we are fully aware of inherent dangers of analogies.

ANNs exchange information between neurons frequently and can be divided into two classes based on their connectivity: feedforward and feedback networks. Especially feedback networks exchange lots of information due to their iterative nature. In terms of WSNs this means that sensor nodes need to communicate with each other frequently. As a rule of thumb, the energy consumption of the transmission of one packet equals 1000 arithmetic operations. Hence, sensor nodes will be depleted faster which will reduce network lifetime significantly. Feedforward networks are better applicable since their connections are directed from input layer to output layer. We can see the whole sensor network as a neural network and within each sensor node inside the WSN there could run also a neural network to decide on the output action. Thus, it is possible to envision two-level ANN architecture for WSNs.

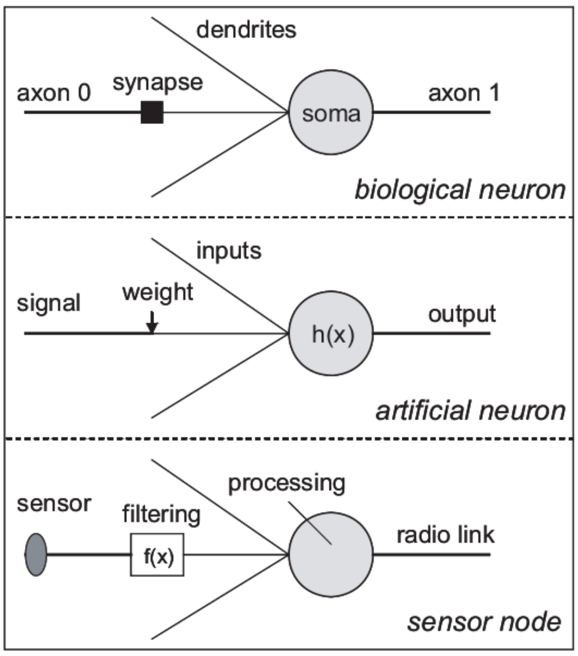

Figure 2.1: Analogy of biological neuron, artificial neuron and sensor node[30]

2.2 Network communication models

Computer networking involves a group of computers communicating with each other for the purpose of sharing information and resources. These computers communicate using a set of communication protocols, mainly the internet protocol suite(TCP/IP) and IEEE 802 standards. These standards are based on the Open Systems Interconnection(OSI) model which is illustrated in Fig:2.2

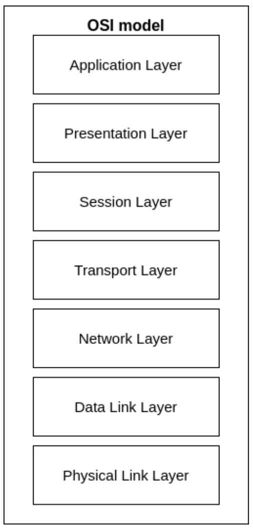

Figure 2.2: OSI model[7]

2.2.1 OSI model

Most networks are organised as a stack of layers with each one being built upon the one below it. The purpose of each layer is to provide services to higher layers above it and provide abstraction [7]. The OSI model consists of seven abstraction layers in which data transmitted from one computer starts from application layer, passes through the physical layer/medium to reach the other computer, and then travels up the stack . The layers in the OSI model have the functionalty as described below::

- **Physical layer:** It is concerned with transmission of raw binary bits(0s and 1s) over a communication channel. The layer defines characteristics such as voltage levels for bits and frequency.
- **Data link layer:** It is accountable for multiplexing data streams, data frame detection, medium access, and error control. It ensures reliable point-to-point and point-to-multipoint connections during a communication. Furthermore, the layer is split into two sublayers. The Medium access control (MAC) layer controls medium access of device to the network. Logical link control (LLC) layer checks for errors and sychronizes frames.

- **Network layer:** It routes packets by discovering the best path across a physical network. The network layer uses network addresses (typically Internet Protocol addresses) to route packets to a destination node.

- **Transport layer:** It breaks the data it into segments on the transmitting end and responsible for reassembling the segments on the receiving end. The layer carries out flow control, sending data at a rate that matches the connection speed of the receiving device, and error control.

- **Session layer:** It is responsible for opening sessions and closing them when communication ends. The session layer can also set checkpoints during a data transfer—if the session is interrupted, devices can resume data transfer from the last checkpoint.

- **Presentation layer:** It defines the encode, encrypt and data compression mechanism between the devices. The layer takes data transmitted by the application layer and prepares it for transmission over the session layer.

- **Application layer:** It is used by end-user applications such as web browsers and email clients.

2.3 Overview of IEEE 802.15.4e

The IEEE 802.15.4e protocol standard maintained by IEEE 802.15 working group, specifies the Medium access control (MAC) sub-layer and physical layer for low-rate wireless private area networks (LR-WPAN). The protocol enables low data rate, low power consumption and low cost wireless communication which typically fulfills the requirements of wireless sensor networks [26]. An example of network stack for IEEE 802.15.4e is shown in Fig:2.3

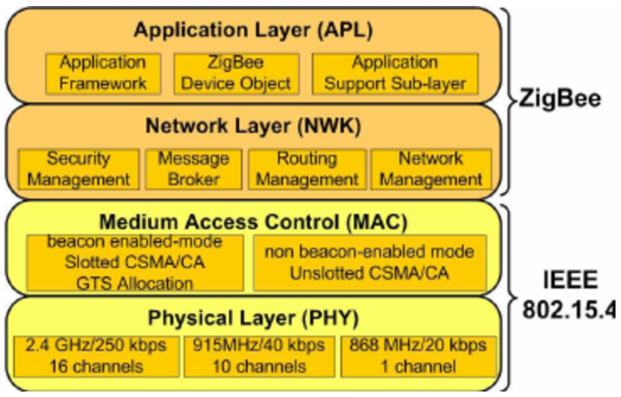
Application Layer (APL)
Application Framework
ZigBee Device Object
Application Support Sub-layer
Network Layer (NWK)
Security Management
Message Broker
Routing Management
Network Management
ZigBee
Medium Access Control (MAC)
beacon enabled-mode Slotted CSMA/CA GTS Allocation
non beacon-enabled mode Unslotted CSMA/CA
Physical Layer (PHY)
2.4 GHz/250 kbps 16 channels
915MHz/40 kbps 10 channels
868 MHz/20 kbps 1 channel
IEEE 802.15.4

Figure 2.3: IEEE 802.15.4e/Zigbee network stack[13]

2.3.1 Network Devices

The IEEE 802.15.4e standard, a LR-WPAN supports two types of devices:

- Full Function Device (FFD): A FFD device can support three operation modes, the device serving as:
  - Personal Area Network (PAN) Coordinator: This device establishes or starts its own network, to which other devices may be associated.
  - Coordinator: This devices associates to a network established by a PAN and provides synchronization services through transmission of beacons. This device does not create its own network.
  - End device: It is associated to a coordinator.
- Reduced Function Device (RFD): The RFD is a device operating with minimal implementation of the IEEE 802.15.4e protocol. An RFD is intended for applications that are extremely simple, such as a light switch or a passive infrared sensor; they do not have the need to send large amounts of data and may only associate with a single FFD at a time.

2.3.2 Network Topologies

Two basic types of network topologies are defined in the IEEE 802.15.4e standard, according to the networking requirements of the applications: the star topology and peer-to-peer or mesh topology. The standard also supports tree topology.

- Star Topology

  In the topology, one unique node acts as a PAN coordinator. The PAN coordinator chooses a PAN identifier, which isn't currently employed by the other network within the sphere of influence. The communication paradigm is that the star is centralized i.e., each device (FFD or RFD) joining the network and willing to communicate with other devices must send its data to the PAN coordinator, which dispatch them to the adequate destination devices.

- Peer-to-Peer Topology

  In this topology, it includes a PAN coordinator, which is nominated, as an example, by virtue of being the primary device to communicate on the channel. However, the communication paradigm within the peer-to-peer topology is decentralized, where each device can directly communicate with the other device in its radio range.

  This network topology enables enhanced networking flexibility, but it induces an extra complexity for providing an end-to-end connectivity between all devices within

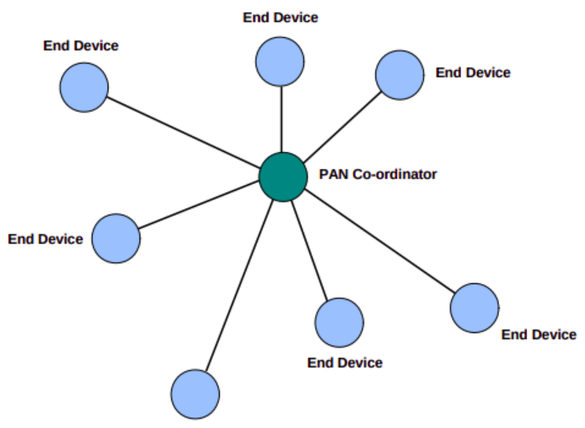

Figure 2.4: Star Topology[8]

the network. The topology allows multiple hops to route data from any device to any other device. However, this complexity of multi hop communication should be defined by network layer of the application and therefore, are not considered in IEEE 802.15.4 specification.

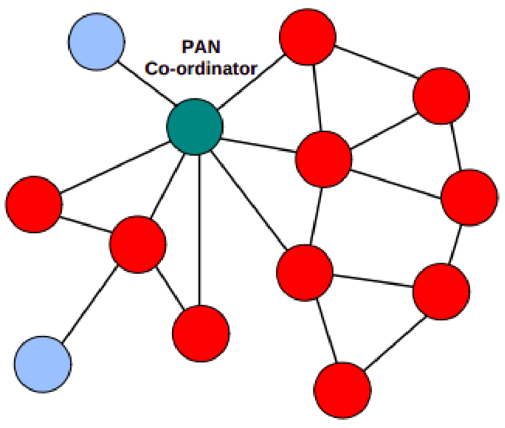

Figure 2.5: Peer-to-Peer Topology[8]

- Tree Topology

  In this topology, it has a structure based on parent-child relationships. Each node (except the PAN Coordinator) has a parent. The node (including the PAN Coordinator) may also (but not necessarily) have one or more children. Each node can communicate only with its parent and its children (if any). Any node which is a parent acts as a local Coordinator for its children. The network can be visualised as a tree-like structure with the PAN Coordinator at the root (at the top).

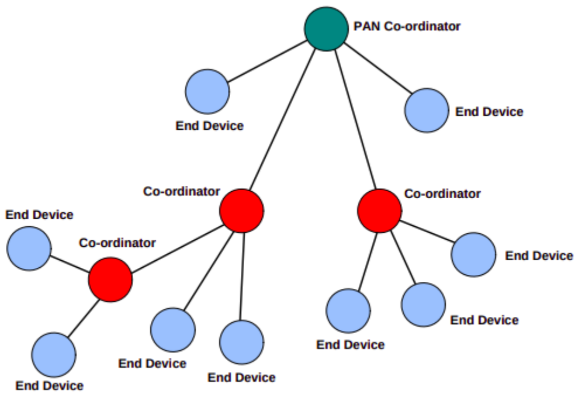

Figure 2.6: Tree Topology[8]

2.3.3 Physical Layer

The IEEE 802.15.4e offers three operational frequency bands: 2.4 GHz, 915 MHz and 868 MHz. There is a single channel between 868 and 868.6 MHz, 10 channels between 902 and 928 MHz, and 16 channels between 2.4 and 2.4835 GHz. The protocol also allows dynamic channel selection, a scan function that steps through a list of supported channels in search of a beacon, receiver energy detection, link quality indication and channel switching. The data rate is 250 kbps at 2.4 GHz, 40 kbps at 915 MHZ and 20 kbps at 868 MHz. Lower frequencies are more suitable for longer transmission ranges due to lower propagation losses. Low rate transmissions provide better sensitivity and larger coverage area. Higher rate means higher throughput, lower latency or lower duty cycles. All of these frequency bands are based on the Direct Sequence Spread Spectrum (DSSS) spreading technique

2.3.4 MAC Layer

The MAC layer provides an interface between the physical layer and network layer. The layer performs the following functions:

- Provides carrier-sense multiple access wit collision avoidance(CSMA/CA) as the access method for the network.
- Provides a reliable connection between devices by using an acknowledgement.
- Generates beacon frame in a coordinator and offers optional guaranteed time slot (GTS) for each device accessing the network.
- Association and disassociation of end device.

2.3.4.1 Time slotted channel hopping (TSCH)

TSCH communications prevents internal interference in the network and improves reliability of wireless communication.

2.4 Overview of Contiki OS and Contiki-NG

Contik-NG [4] based on Contiki OS[5] is one of multiple active open-source IoT OSs that exist. FreeRTOS[1], and RIOT[6] are other open-source examples. Contiki was originally developed at RISE SICS, and Contiki-NG is maintained and further developed at RISE SICS. Its broad use in both commercial and scientific projects, it is the IoT OS used as tool during this project. The following section will discuss the inner workings of Contiki-NG.

2.4.1 Features of Contiki-NG

Contiki-NG focuses on standardized communication technologies for modern IoT platforms by modernizing the structure, improving the documentation, and implementing a more agile development process compared to that of Contiki. There are several features that make Contiki-NG stand out:

- **Hardware support:** Contiki-NG currently supports over eight hardware platforms: TI CC2538 development kit, NXP jn516x series, Nordic Semiconductor nRF52 development kit, OpenMote cc2538, Tmote sky, TI CC26x0 and CC13x0 platforms, and Zolertia Zoul platforms. An even wider range of hardware devices built on top of these platforms. Furthermore, new platforms can be supported easily when required.

- **Libraries:** Libraries are available to the user to make building applications easier. Examples of this are: circular buffers, queues, linked lists, etc..

- Memory management: Contiki-NG's memory footprint can be configured as low as 10 kB. To manage low memory footprint, a library is available that frees up memory blocks, while maintaining an overview of where memory is then saved.

- **Energy efficiency:** Contiki-NG is capable of saving power. Radio duty cycling is a big aspect of this as it presents control over the most power intensive part of an IoT device. Power consumption can also be measured which is useful when carrying out experiments, or debugging unexpected battery drainage.

- **Developing tools:** Contiki-NG's developers workflow simplifies application development significantly. Inside the application's "Makefiles", modules can easily be added or removed. The build system speeds up compiling by making it unnecessary to recompile code that has not been changed. Applications are easily uploaded to different hardware boards using "make" commands. A logging module is available to easily debug applications, a shell module can be used to change node configurations during runtime, and an active community can answer specific questions or help solve problems.

2.4.2 Contiki-NG architecture

Contiki-NG's basic architecture can be divided into four basic blocks: the hardware platform, an hardware abstraction layer, the actual OS, and the top layer for Contiki-NG application programs. Contiki-NG's repository consists out of 5 main folders and can be mapped onto the architecture in the following way:

- **arch:** contains the hardware-dependent code like CPU and radio drivers. It directly maps with the hardware abstraction layer of the Contiki-NG architecture.
- **os:** has all the OS code and aligns with the OS layer. Parts of the OS code are: primitives, the storage file system, libraries and services built on top of the primitives.
- **examples:** the example folder contains application programs that can be directly up- loaded onto hardware and are used to tutor new users on the different features and protocols of Contiki-NG. It is shown in the top left corner of the architecture inside the applications layer.
- **tools:** apart from the firmware of Contiki-NG, the repository provides useful tools to flash boards for the first time, do network simulations and provide serial logging.

2.4.3 Primitives of Contiki-NG

Contiki-NG's OS and applications are built on top of a set of primitives. Processes and protothreads are the most important ones. To understand these, first the concept of execution contexts has to be explained. Code run in Contiki can run in two execution contexts: cooperative or preemptive, as shown in Fig:2.7. Cooperative code runs sequentially, while preemptive code interrupts the cooperative code and does not let the cooperative code continue until completion of the preemptive code.

2.4.3.1 Processes

All Contiki programs consist out of processes and they a run in a cooperative execution context. In Fig:2.7, at t_0, a cooperative "Process A" starts. At t_1 there is a switch to another cooperative "Process B" with some overhead. Once an interrupt occurs, a switch is made to a preemptive scheduling context. Only after the interrupt finishes, Process B" resumes and continues till t_3. Another preemption occurs during execution of "Process D", when a real-time timer preempts the current process. A process is a code block executing periodically. Usually one process is started at boot-up, and other processes can start when either a timer expires, or an event takes place. Given no interrupts take place. A Contiki process consists of two parts: a control block and a thread. The control block is declared at the top of a program containing the name, a pointer to the thread, and the state:

```
PROCESS (example_proc , "Example ");
AUTOSTART_PROCESSES (&example_proc );
```

Where the code between macros PROCESS_BEGIN(), PROCESS_END() contains the thread:

```
PROCESS_THREAD (test_proc , ev , data)
{
PROCESS_BEGIN ();
printf (" Hello world !");
}
PROCESS_END ();
```

Here the control block starts the process automatically using: AUTOSTART_PROCESSES(), otherwise the process would need a manual start using process_start() .

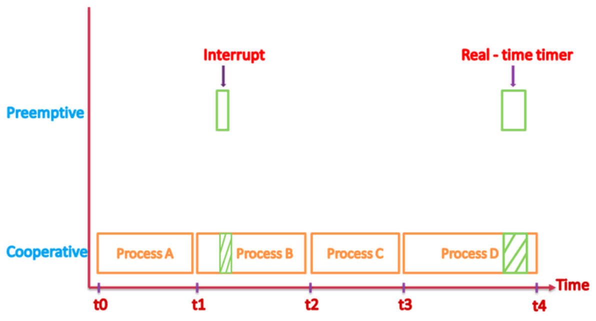

Figure 2.7: Scheduling in Contiki[25]

2.4.3.2 Protothreads

Unfortunately, using processes in Contiki is not straightforward. Processes make use of a lower level programming abstraction made for Contiki called protothreads. Most IoT OSs use a so called event-driven programming model; where the flow of the program is determined by timers, sensor triggers, etc. This keeps memory overhead low, but makes programming often tedious, and difficult to understand. Reason is the state machine coding style needed for blocking sequences. Protothreads are also event-driven, but provide sequential control by having them run on the same stack and switch context by rewinding the stack. This is achieved with negligible RAM overhead, and makes coding clearer using less lines. However, a disadvantage is that variables are not guaranteed to maintain values when switching processes due to their stackless nature. Manual blocking needs to be applied using global variables to provide guarantees.

Chapter 3

Literature study

The aim of this literature study is to understand the existing technology used to achieve AI on low power devices and also different methods of indoor localisation.

3.1 AI on low power devices

The use of NN models in low power, low resource microcontrollers (μCs) is not a new field. The characteristics of low power devices of limited computation and less memory poses challenges to deploy AI models. However, in [41] offloading to cloud is done and a CNN-based object-inference task is implemented on a Nvidia Jetson TX1 device. This implementation is less expensive than local execution but high variability latency of 2-5s is observed. Hence, offloading to the cloud is not a viable solution to implement real time tasks like security camera–based object-recognition which usually require a detection latency of less than 1s to capture and respond to target. The memory occupancy and cost of computation depends on the design of the model, number of neurons, layers and inputs. In [34], the paper describes an implementation of neural network on a low end device. An ANN is trained with the back-propogation (BP) algorithm is validated using a low-end and inexpensive PIC16F876A 8-bit μC. The ANN is modularized in matrix form, simplifying and modularizing all the feedforward and backwards propagation. The results obtained show less inference latency and proves compatibility of embedding ANN on low power and low cost devices.

An ANN with a BP training algorithm and a single hidden layer is used to implement a hurdle-avoidance system controller using a AT89C52 μC in [18]. The authors describe the design of neural network based intelligent autonomous vehicle. Two neural network controllers namely hurdle avoidance and goal reaching are constructed to accomplish the navigation task. Both these controllers are feed forward neural networks trained off line with back propagation learning algorithm and implemented in real time with AT89C52 micro-controllers.

According to [42], wireless sensors are troubled with constrained energy and computational ability. It is difficult in such wireless sensor networks (WSNs) to apply traditional data fusion methods. The authors propose a WSN with distributed data fusion which uses a neural network. Sensors are randomly deployed in a field and divided into several squares with each representing a sub domain of sensor field. The sub domains select a cluster head (CH) which has a pre-trained neural network that collects sensor data from its neighbours to estimate the possibility of forest fire. If the possibilty exceeds a certain value, the CH transmits the data to sink for further estimation. The presented schema is simulated and results show that the method increases the lifetime of WSN.

In a smart home environment that has sensors connected with a network, certain events such as detecting fire and triggering an alarm are critical demanding to detect fast while other events like starting the heating system and closing off water supply when a leak is detected are non-critical requiring reliable response. The authors of [15] craft a local little neural network that offers fast response and also acts a filter to limit the number of inputs to the bigger neural network in the cloud. Fig:3.1 shows the architecture of the system in which the little neural network classifies only a subset (2) of the N output classes. Small computers such as Raspberry Pi®and Intel®Edison with processing speed over 400MHz and with 1GB RAM are used for training an executing local neural networks.

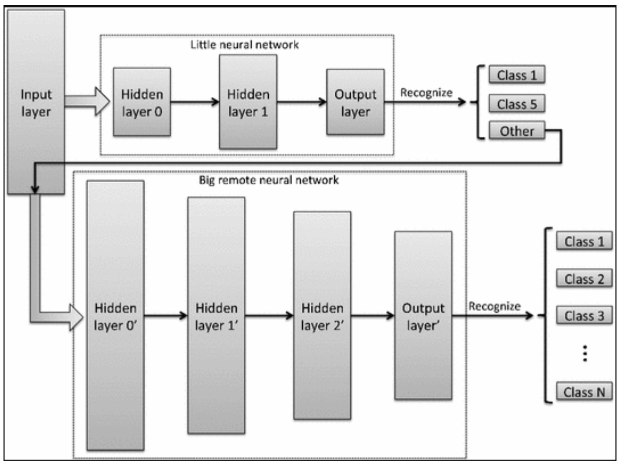

Figure 3.1: Architecture of Big-Little feed-forward neural network[15]

3.2 Indoor Localisation

Indoor localisation nowadays has become a major focus in any kind of industry. From collective sports to healthcare and manufacturing, all are looking for positioning systems allowing to optimize their core business and maximize their performance. To meet these requirements there are various methods implemented to estimate and enhance indoor positioning. Owing to the sufferance of global navigation systems in indoors areas many researches have developed wireless fidelity (Wi-Fi) based localisation systems. These systems rely on various measurements such as Time-Of-Arrival (TOA), Time-Difference-Of-Arrival (TDOA), Angle-Of-Arrival (AOA), and Received Signal Strength (RSS) of Wi-Fi signal.

In [9] trilateration and fingerprinting method which use RSS have been compared. Trilateration calculates distance by using RSSI values depending upon the inverse relationship of distance and signal strength. In Fingerprinting there are two phases, online and offline. In offline phase the RSSI of access points at selected positions in the region of interest is collected and is labelled with correct location and a database is formed. In online phase RSSI of access points is collected and compared with the database to find the target location. According to [9] fingerprinting has less average error, needs less access points and is cost effective when compared to traditional trilateration method. However, this approach requires the access points to be placed at fixed points and requires them to be mains powered.

In [39] fingerprinting localisation for indoors is done using beacons. It uses bluetooth low energy (BLE) beacon-based fingerprinting, where the RSSI of beacons at the predetermined location and the location coordinates are stored in a database as reference points. This method does not require line-of-sight measurements of access points (APs), has low complexity, and gains high applicability in the complex indoor environment.

In [22], the authors propose a improved RSSI based indoor localisation method. Here a system of three anchor nodes and one target node is realized. The relative position of target node is estimated. This process includes three phases: data acquisition, distance estimation and position computation. In the first phase data from different sensor node is collected and sent to sink node. Secondly distance between target node and anchor nodes is computed using range based techniques like received signal strength (RSS) considering the nodes in line of sight (LOS). Then traditional methods like trilateration, multitrilateration or triangulation method is used to localise the target node. This method claims to have reduced error in trilateration based localisation technique in WSN, using RSS of anchor nodes.

Indoor positioning done using Wi-Fi signals has a major disadvantage of multipath propagation which distorts these signals, leading to an inaccurate indoor localisation. In order to solve this, the authors in [43] improve the localisation accuracy by using an approach consisting of usage of fingerprints based on channel state information (CSI). The proposed approach has three phases. In the first phase, a model is created for the fingerprints of the environment which is considered to perform localisation. This model allows to obtain a precise interpolation of fingerprints at positions where a fingerprint measurement isn't available. In the second phase, considering only the fingerprint measured at the receiver's location this model is utilized to obtain a preliminary position estimates. Finally, in the third phase, this preliminary estimation is combined with the dynamical model of the receiver's motion to get the final location estimation.

Owing to the higher performance of AI based indoor localisation techniques than the traditional methods in [38], the authors propose improved data augmentation technique based on received signal strength indication (RSSI) values for fingerprint indoor positioning systems. Here the augmented RSSI data was generated and the fingerprints were constructed. This method proves to have better accuracy than the CNN model localisation accuracy.

Three localisation schemes that are based on different types of neural networks are compared in [36] which also involves comparison of memory requirements for computation. This study considers WSN localisation using ANN, radial basis function (RBF), and recurrent neural networks(RNN). The ANN used is similar to the neural network used in this thesis which implements a two layer network using two sets of nested loops. The complexity is $O(m^2)$, where m is the greatest number of nodes in the two layers. According to the paper, the ANN uses the least amount of memory and minimum compared to RBF and RNN. This provides a good motivation to use MLP in our thesis as we are aiming towards minimum memory based approach along with the overhead of operating system and network stack memory requirements.

3.3 Summary

In [34], [18], [42], [15] an ANN is deployed on a low power device for applications such as threat detection and hurdle avoidance. Thus, AI on low power devices can be achieved by designing compatible model. With required number of inputs, neurons and layers an ANN can be successfully deployed on a low power device and the objectives of low latency with less computation and memory can be achieved. ANNs are commonly used as the classifier after applying suitable non learnable feature extractors due to their relatively moderately resource demand. In fact, ANNs are much less demanding than deep convolutional networks in terms of memory and computing requirements. For localisation RSSI based methods are most suitable as RSSI requires no additional hardware and can be found on any device utilizing any type of wireless technology.

In the above proposed localisation methods, the devices involved to perform localisation are not wirelessly connected to each other. They do not collaborate together in estimating the location. This thesis focuses on optimising ANN for low power devices that collaborate with each other by forming a wireless network to implement the inherent non -linear function of localisation.

Chapter 4

Sytem Design

The indoor localisation approach in this thesis uses the received signal strength indicator (RSSI) from the signals broadcasted by the target device whose location needs to be determined. The RSSI values observed by the nodes deployed on the ceiling perform indoor localisation using two methods: central localisation method and distributed localisation method. For both the methods, the approach begins by collecting samples of the RSSI values at certain known locations. The samples are used to train the model parameters of a neural network and later deployed in the node(s). Next, a node selected at random forms a IEEE 802.15.4 based hierarchical network with other nodes deployed on the ceiling. Localisation is then performed by matching the observed real-time RSSI vector with the collected samples using machine-learning techniques. This process is divided into three phases: fingerprinting phase, training phase and estimation phase.

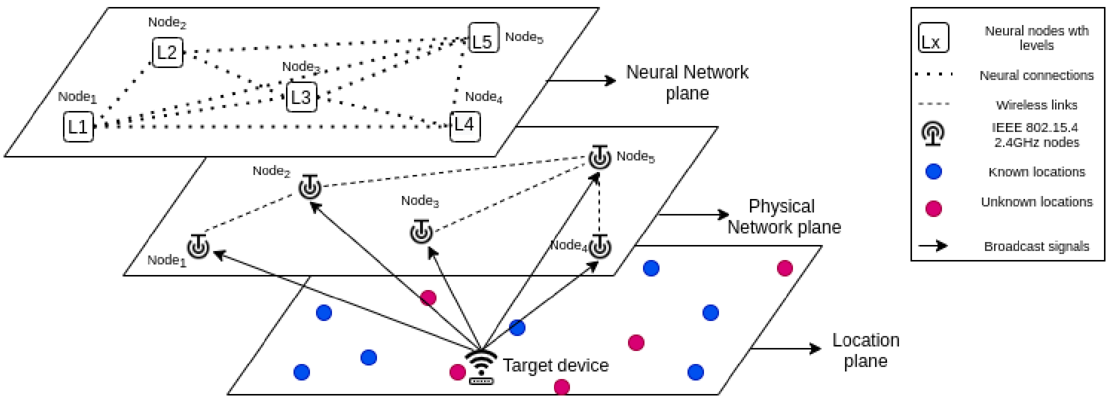

Figure 4.1: AI based localisation

The Fig. 4.1 shows the planar structure of localisation system. The physical network plane shows the random deployment of wireless nodes that form a wireless network. The neural network plane achieves the localisation of target device.

4.1 Overview

Central localisation method

In this method a single output node contains an artificial neural network (ANN) that predicts the location of target device using RSSI measurements. The node takes the measured values from itself and from all other nodes which can be called as input nodes. Specifically, each neuron in the input layer maps to each of the nodes and hence the size of input layer depends on the number of nodes deployed on the ceiling. In order to send the measured values, each input node needs to be connected to the output node or a network needs to be formed. The process of network formation is explained later in this chapter.

Distributed localisation method

In this method every node has an ANN that predicts the location of the target device. Each node takes the measured RSSI values from itself and from all other nodes. In addition, each node also takes the predicted location values from all the other nodes. The output of a node acts as an input to other nodes. Specifically, each measured RSSI value and output values of a node is mapped to neurons in the input layer of the ANN in all the other nodes. Output value of any node which takes both types of values (RSSI and predicted values) as inputs can be chosen as final output value.

4.1.1 Fingerprinting phase

In this phase, the fingerprinting database is constructed with one or many parameters. Assume the localisation area consists of known locations p_i=(x_i, y_i), where i = 1, 2, ..., n, n is the total number of known locations.
A fingerprinting database is expressed by:

$$T = (f_1, p_1), (f_2, p_2), ..., (f_n, p_n) \tag{4.1}$$

where f_i represents the fingerprint pattern for i^{th} location and can be made of one or several parameters.

$$f_i = [m_{i1}, m_{i2}, ..., m_{ij}] \tag{4.2}$$

where m_{ij} represents the different fingerprints of p_i and j is the number of nodes.

Central localisation method

In this method, each node on the ceiling measures RSSI value from the signals broadcasted by the target device which is placed at each known location p_i. Let rss_{ij} refer to the RSSI values from j^{th} node at p_i. The definition (4.2) becomes f_i = [rss_{i1}, rss_{i2}, ..., rss_{ij}]. The fingerprint pattern f_i and its corresponding location p_i is stored in a database as shown in Fig. 4.2. This is repeated for the n known locations. The structure of the database is shown in Table 4.1

Corresponding Fingerprint	Location
$f_1 = [rss_{11}, rss_{12}, ..., rss_{1j}]$	$p_1 = (x_1, y_1)$
$f_2 = [rss_{21}\ rss_{22}, ..., rss_{2j}]$	$p_2 = (x_2, y_2)$
.	.
.	.
.	.
$f_n = [rss_{n1}, rss_{n2}, ..., rss_{nj}]$	$p_n = (x_n, y_n)$

Table 4.1: Structure of database for central method

Distributed localisation method

In this method, each node on the ceiling measures RSSI value from the signals broadcasted by the target device which is placed at each known location p_i. Also each node computes the location $r_i^j = (x_i^j, y_i^j)$ of the target device placed at the known locations. The coordinates r_i^j are assumed to be computed via other indoor positioning techniques by each node. To simplify, we also assume that computed coordinates are highly accurate. The definition (4.2) becomes $f_i = [rss_{11}, rss_{12}, ..., rss_{1j}, r_1^1, r_2^2, ..., r_i^j]$. The fingerprint pattern f_i and its corresponding location p_i is stored in a database as shown in Fig. 4.2. This is repeated for the n known locations. The structure of the database is shown in Table 4.2

Corresponding Fingerprint	Location
$f_1 = [rss_{11}, rss_{12}, ..., rss_{1j}, r_1^1, r_1^2, ..., r_1^j]$	$p_1 = (x_1, y_1)$
$f_2 = [rss_{21}, rss_{22}, ..., rss_{2j}, r_2^1, r_2^2, ..., r_2^j]$	$p_2 = (x_2, y_2)$
.	.
.	.
.	.
$f_n = [rss_{n1}, rss_{n2}, ..., rss_{nj}, r_n^1, r_n^2, ..., r_n^j]$	$p_n = (x_n, y_n)$

Table 4.2: Structure of database for distributed method

4.1.2 Training phase

This phase consists of modelling the input vector and the output vector of the fingerprinting database. Modelling involves training of a network based on samples of the known database. As MLP/ANN can be deployed on low power devices with low memory footprint, neural network technique is proposed to predict the location. This technique is adopted for both the methods. Neural networks are usually used to model the complex relationship between the input vector and the output vector, which makes it suitable for the fingerprinting based localization technique.

In this thesis, we use the feed-forward network which allows the signals to travel from input to output without feedback and tends to be straight forward network that associates

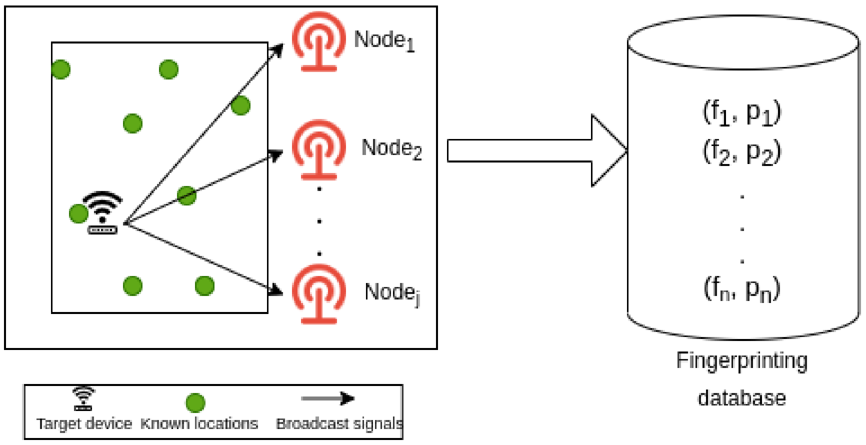

Figure 4.2: Fingerprinting methodology

inputs with outputs. The behaviour of the neural network depends on both the input-output transfer function and the appropriate weights of the neurons which can reduce the error between desired outputs and real outputs. During the learning process, the sigmoid activation function is used to translate the input signal to outputs signal. More importantly, the network needs to be trained according to the specific task which involves setting the weights on the connections appropriately to reduce the error between desired output and real output. We use backpropagation algorithm [32] to train the network. In Fig.4.3, a simple neural network with the fingerprint pattern f_i with its RSSI values placed in parallel as inputs and the 2D coordinates of the location as outputs is shown.

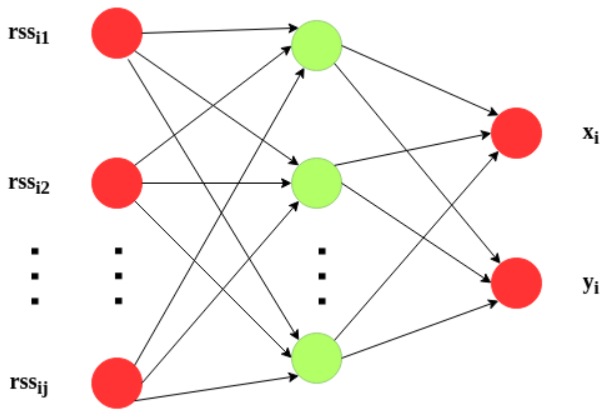

Figure 4.3: Neural network architecture

4.1.3 Estimation phase

Central localisation method

In this phase, the targeted user's location is estimated. It includes 3 main stages:

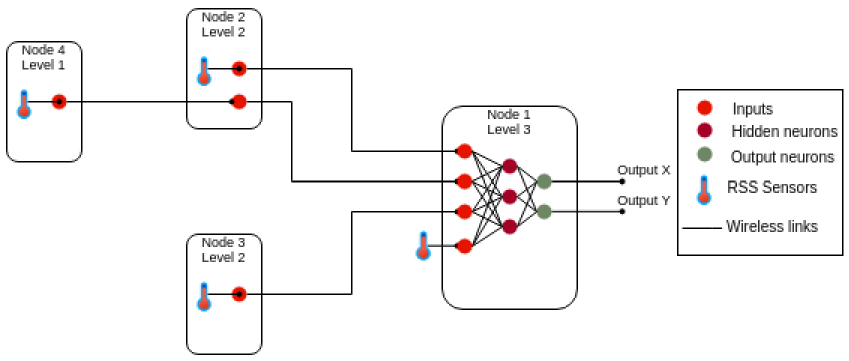

Figure 4.4: Connections in Central method with 4 nodes

1. Formation of a hierarchical network of N levels with j the nodes.
2. Communication between the nodes.
3. The trained feed-forward neural network in the Nth level node estimates the location based on the inputs received from the network.

A hierarchical wireless network is formed where each node is assigned a level with output node being at the highest level. Fig. 4.4 shows an example of network hierarchy with 4 nodes. The depth of the level and network topology depends on the number of surfaces on the ceiling and capacity of the output node in terms of energy budget. The output node is chosen at random which begins the formation of network by broadcasting beacons periodically. All other nodes act as input nodes that begin scanning for beacons. Upon successful scan, a node sends an association request to join the network and after joining, a level is assigned to the node. Now this node will in turn will broadcast beacons and assigns a level to the new nodes joining the network through it. This process repeats until all the available nodes join the network. Once the network is formed each node maintains a neighbor list with node ID and level information of its immediate neighbors. In terms of data routing, as RSSI values are directed towards the output node, each input node at a level L sends its data and also forwards data received from lower levels to the node present at L+1 level in its neighbor list. If the star network is formed, the routing is simple as there are only 2 levels. However, in case of tree network, the routing involves multi-hop data communication as there are more than 2 levels.

The localisation process begins when the target device whose location needs to be determined begins transmitting broadcast signals from unknown location as shown in Fig. 4.5. Each node on the ceiling measures the RSSI values from signals. The input nodes forward the values to the output node with routing method as described above. Once the output node receives enough values according to its input layer size, the trained FNN estimates the location and it is considered as final output.

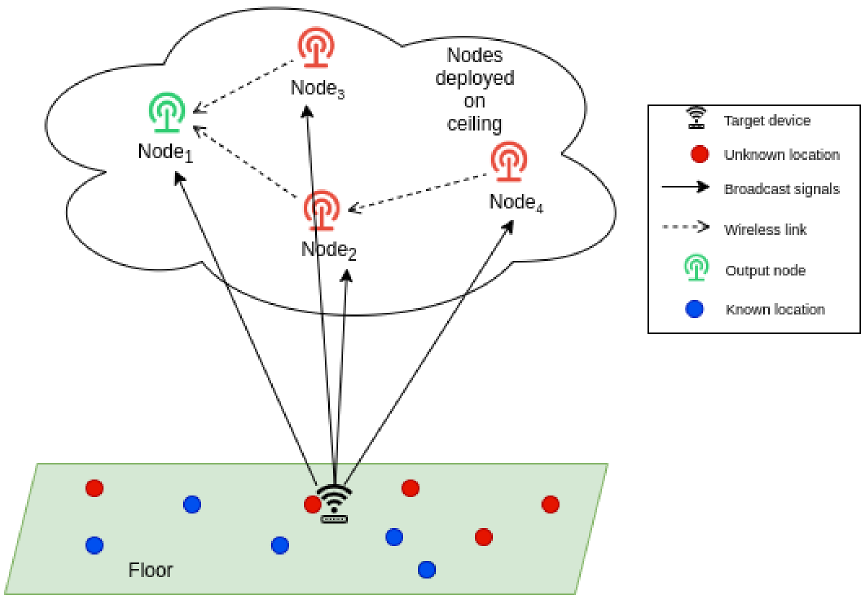

Figure 4.5: Central localisation method with 4 nodes

Distributed localisation method

In this phase, the targeted user's location is estimated. It includes 3 main stages:

1. Formation of a hierarchical network of N levels with the j nodes.
2. Communication between the nodes.
3. The trained feed-forward neural network in the all the nodes estimates the location based on the inputs received from the network.

The depth of the level depends on the number of surfaces on the ceiling and capacity of the each node in terms of energy budget. A node is chosen at random which begins the formation of network by broadcasting beacons periodically. All other nodes begin scanning for beacons. Upon successful scan, a node sends an association request to join the network and after joining, a level is assigned to the node. Now this node will in turn will broadcast beacons and assigns a level to the new nodes joining the network through it. This process repeats until all the available nodes join the network. Once the network is formed each node maintains a neighbor list with node ID and level information of its immediate one hop neighbors. In terms of data routing, as RSSI values and output values of each node are sent to all other nodes, a node at level L sends its data and forwards data to its one hop neighbors. Each node stores its measured RSSI values and values received from the network via its immediate neighbors. Once the node receives enough values according to its input layer size, the trained FNN estimates the location and the output is sent to other nodes. This method has a drawback, when the target user broadcasts its signals, each

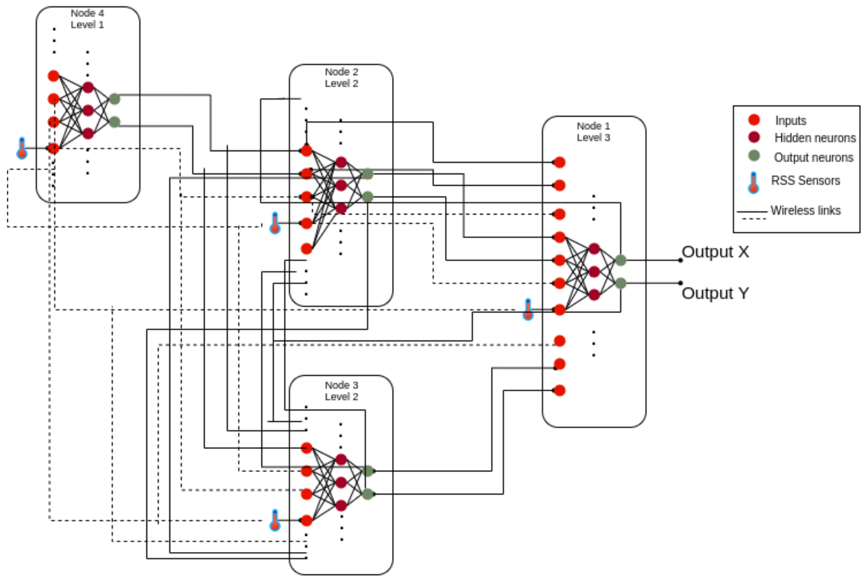

Figure 4.6: Connections in Distributed method with 4 nodes

node takes measured rssi values from itself and from all other nodes. However, each node also waiting for output values of all other nodes to compute its output which results into a deadlock. To overcome this problem, the nodes at level 1 take only RSSI values as inputs and compute their outputs which is propagated into the network. Now, nodes in higher levels receive both types of values and compute their outputs.

The localisation process begins when the target device whose location needs to be determined begins transmitting broadcast signals from unknown location as shown in Fig. 4.7. Each node on the ceiling measures the RSSI values from the signals. As mentioned above each node at level 1 stores RSSI values measured by itself and from other nodes. Once the nodes receives enough values according to their input layer size, the trained FNN in each node estimates the location. These predicted values are now propagated to the nodes at level 2 and above. Nodes at these higher levels store RSSI and predicted location values. Once each node receives enough values according to its input size, the trained FNN predicts the location of the target device and sends it to other nodes. This location predicted by any of these nodes can be considered as final output. In this thesis, we consider the output of the node which forms the network and has the highest level as final output.

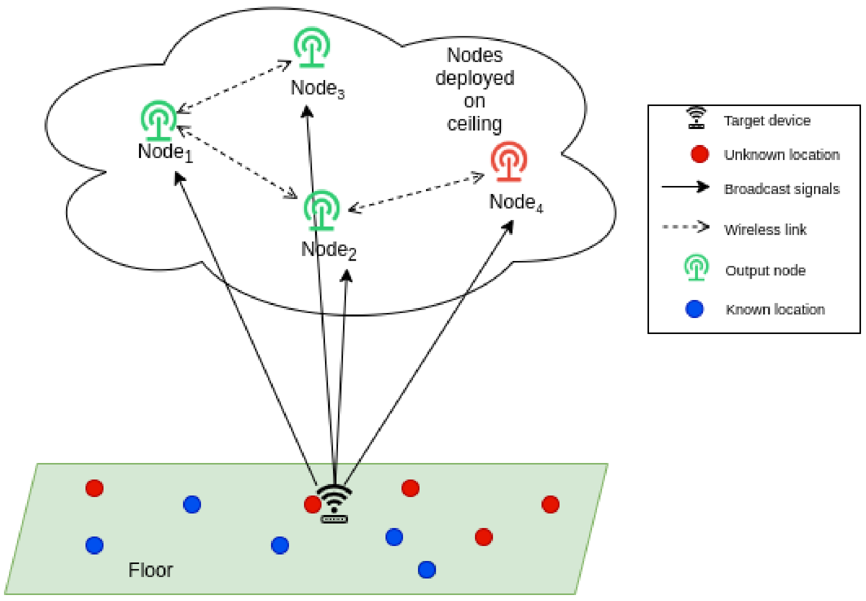

Figure 4.7: Distributed localisation method with 4 nodes

4.2 Design adaptation to Low power device

Low power devices have a limited energy budget, memory and computational resources which limits its availability for applications that use NNs to solve non linear approximations as they require massive computational resources and memory. However, over the last few years a lot of research has been to done towards optimised inference algorithms to run NNs on power constrained devices with acceptable performance in terms of accuracy and speed [11]. On the hardware side, to improve the performance of low power processors, researchers are proposing parallelism, memory access optimisation, near-threshold technology and low-power fixed function hardware accelerators [44]. On the software side, it is shown that in many application scenarios, multilayer fully connected networks on embedded devices are just as effective as done in cloud computing [28], [31], [29], [15].

In order to successfully execute the NN on a low power device, the network needs to be optimised in terms of its size, training methods (e.g. smaller training data sets), connections between neurons in the network, activation function, etc. In this thesis we focus on optimising the size of NN model, especially the memory occupied by the network. A feed-forward NN system includes NN model with optimised weights and the inference algorithm that makes the prediction based on inputs. We assume that both occupy equal amount of memory. Let E_m be the memory occupied by NN model and E_a be the memory occupied by the inference algorithm. The total memory (E_t) occupied is given by, $E_t = E_m + E_a$ and E_m is given by:

$$E_m = (L_{databuffer} + N_{neurons} + N_{weights}) \times sizeof(datatype) \tag{4.3}$$

where $L_{databuffer}$ is the buffer length of one input sample, $N_{neurons}$ is the total neurons including biases seen as an additional neuron to every layer and $N_{weights}$ is the total number of weights in the network.

Chapter 5

Implementation

5.1 Hardware Platfrom

5.1.1 Zolertia Firefly

The wireless node is the central element in the wireless neural network. Communication and processing takes place in the nodes. The node stores and executes the communication protocols and the data processing neural algorithm. Figure 5.1 shows the overview of the node architecture. This thesis uses the Zolertia Firefly platform[3] which is used for

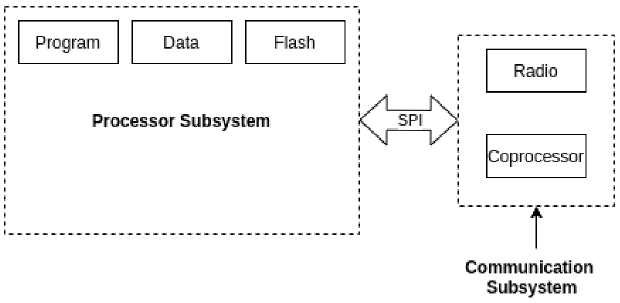

Figure 5.1: Neural node architecture [14]

developing wireless sensor network applications. The node has Texas Instruments based CC2538 system-on-chip(SOC) which includes an 32 bit ARM Cortex-M3 microprocessor with 512 KB of flash memory and 16 KB of SRAM. The processor core provides low-power consumption with integrated sleep modes. The SOC has an integrated low-power communication subsystem that has IEEE 802.15.4-compliant radio transceiver. The command strobe/CSMA-CA processor(CSP) provides the control interface between the CPU and radio.The radio subsystem provides a transmission rate of 250 kbps over 16 channels in the 2.4-GHz band.

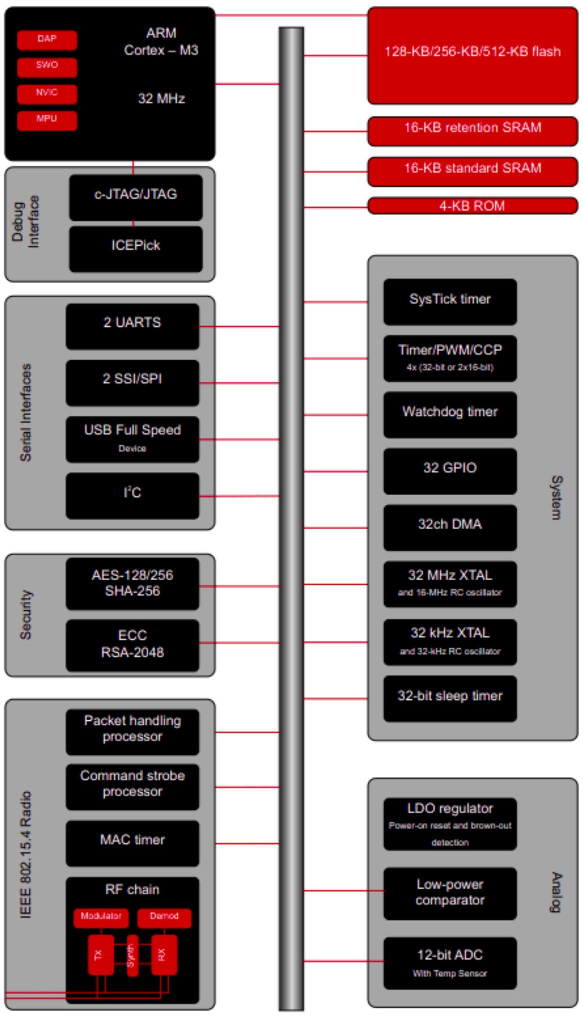

Figure 5.2: CC2538 architecture[21]

5.2 Test bed

The nodes were deployed in a lab (7m x 7m) in the building of Electrical Engineering department of the Eindhoven University of Technology which was used as the test room. This lab has cable gutters mounted on the ceiling. On these cable gutters, 23 firefly nodes [3] are mounted on Beaglebone®(BB) boards and placed at random positions as shown in Fig:5.4. The BBs form a wired mesh connection which are connected to a server which runs the gravel network [2]. On the lab floor, 31 training locations/known locations are marked which are determined in reference to the two orthogonal walls. Also, 10 unknown positions are marked as test locations as shown in Fig. 5.3. The target device, an firefly node whose location needs to be determined was placed on a tripod and the antenna of node facing parallel to the ceiling and placed at an height of 1.2m from the floor.

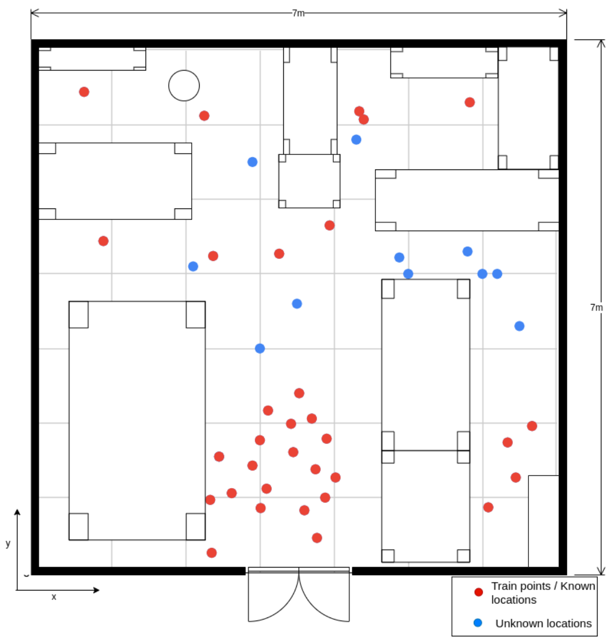

Figure 5.3: Lab floor layout

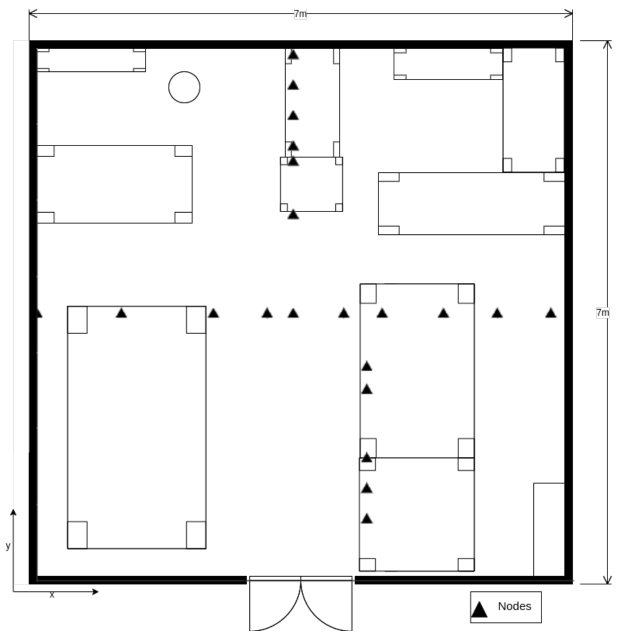

Figure 5.4: Deployment of nodes on the ceiling

5.3 Contiki-NG OS

The nodes need to communicate in a synchronised way as the data being transmitted cannot be lost or corrupted due to interference in the network or have many re-transmissions.

Contiki-NG comes with a rich network-stack features to allow communication between the nodes and uses the traditional OSI stack to implement its network protocol stack, called NETSTACK which is shown in Fig. 5.5 The NETSTACK implements the following

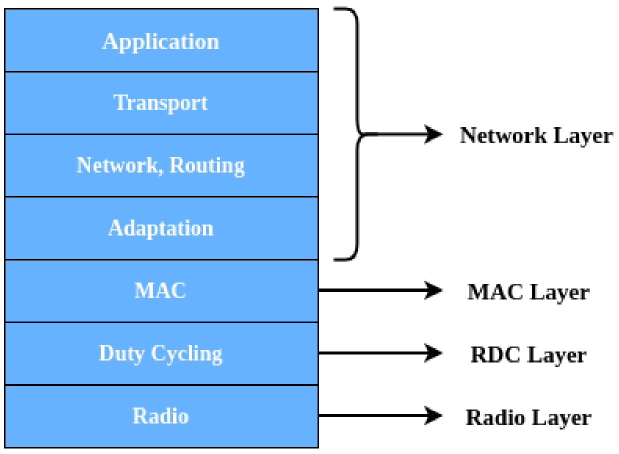

Figure 5.5: Network protocol stack of Contiki-NG [35]

four layers:

- **Radio Layer:** It is the lowest layer in the NETSTACK and is handled by radio module of the node. Most radio radio layers work on the IEEE 802.15.4 protocol mechanism.
- **RDC Layer:** The Radio Duty Cycling (RDC) layer saves energy by allowing a node to keep its radio transceiver off most of the time.
- **MAC Layer:** Contiki-NG offers implementation of protocols such as CSMA and TSCH for IEEE 802.15.4e. The CSMA implemented is non-beacon enabled and its unslotted which does not meet our requirements. TSCH is a MAC layer protocol which offers global synchronization and nodes communicates with each other at fixed time slots. These slots are usually 10ms long with each of them either dedicated for transmitting, receiving or sleeping. The time slots are grouped into slotframes of specific length which repeat periodically. The protocol also facilitates channel hopping, which means a node transmits packets to its destination at different frequency channels for different occurrences of the same slot.

  The MAC layer in Contiki-NG also implements and manages node associations. By configuring parameters, a node can be chosen as a PAN Coordinator which starts the formation of network and other devices join the network after hearing an Enhanced Beacon (EB) from other nodes. Scheduling is important for TSCH based protocols,

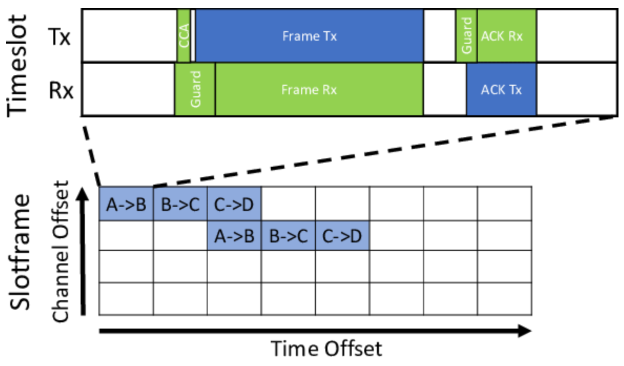

Figure 5.6: TSCH timeslot and example slotframe[17]

which determines which how the slots are configured and distributed in time on any node. By default, Contiki-NG has two types of scheduling:

– **Simple minimal schedule:** This schedule provides a single time-slot for all communications between the nodes. Basically it is contention based schedule with one time-slot. This is inefficient as all nodes compete to transmit and receive at the same time and thus interference is high, especially if there more nodes in the network.

– **Orchestra:** The Orchestra [16] enables nodes to compute their own local schedules based on the network dynamically. The scheduler works along with a routing protocol called RPL providing four common types of slots: common shared slots, sender based shared slots, receiver based shared slots, and sender based dedicated slots.

The proposed design explained in Chapter 4 does not specify the exact topology or number of neighbors per node. This means the load on the nodes will not be uniform and depending on the topology formed, certain nodes will have to listen to many messages from its neighbors and forward them to next nodes which means there are hotspots where data traffic is high and varies. Orchestra computes schedules statically that does not adapt to varying traffic. TRaffic-aware Energy Efficient (TREE) [27] scheduler overcomes the limitations of Orchestra. The algorithm handles communication timeslots (cells) in a distributed way regardless of any traffic pattern or routing topology. TREE monitors the incoming and outgoing traffic with its direct neighbors and transmission acknowledgments. It coordinates with its direct neighbors to prevent resource allocation conflicts. Every device in the network monitors outgoing packet queue towards a specific neighbor and when queue is fully loaded it triggers the allocation of a new random cell for that neighbor. The algorithm also monitors cell activity. If the cell is not being used or if there any interference for that cell, cell deletion is triggered.

- **Network Layer:** Contiki-NG implements the uIP low-power IPv6 stack. The layer contains two sublayers: upper IPv6 layer and the lower adaptation layer. These layers run on top of the MAC layer. In terms of routing, the OS implements Routing Protocol for Low-power and Lossy Networks (RPL) as specified in RFC 6550 [45]

5.4 Memory Constraint

Contiki-NG implements IPv6 network stack with 6LoWPAN as adapation layer working on top of IEEE 802.15.4 MAC and RPL routing protocol to support IPv6 seamless operation on low power devices. However, the memory required to implement the stack exceeds the memory of Zolertia Firefly which is 16 KB as shown in Table 5.1. The .text is the code segment stored in flash memory, .data and .bss are the uninitialised and initialised data stored in the RAM.

.text (bytes)	.data (bytes)	.bss (bytes)	dec (bytes)
123548	3236	20040	146824

Table 5.1: Minimum static memory occupied by IPv6 network stack on Contiki-NG OS

The memory constraint of Zolertia Firefly pushes us to further implement a very light network stack. In this thesis, we implement a custom network layer on top of the IEEE 802.15.4 TSCH-MAC with TREE scheduler which is already implemented in Contiki-NG.

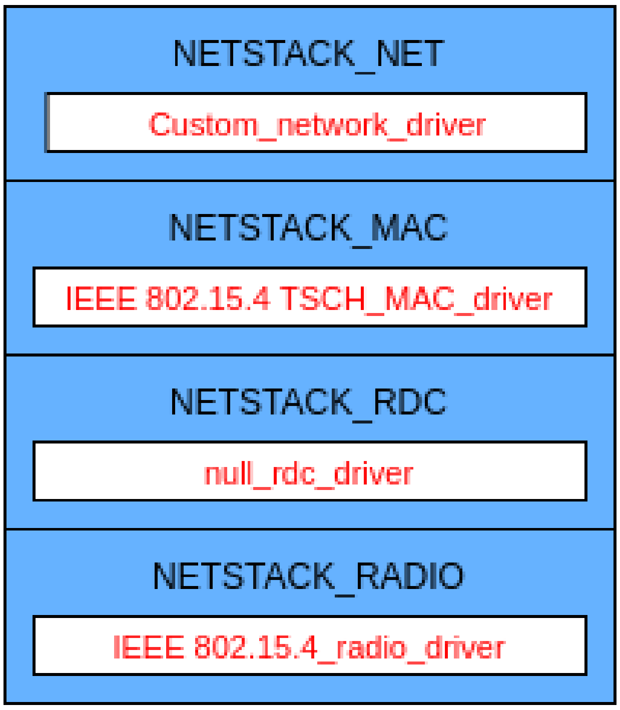

Figure 5.7: Network stack with custom network layer

5.5 Custom Network Stack

As mentioned earlier, Contiki-NG's NETSTACK organises the network modules into a complete protocol stack covering all the traditional OSI layers. Since this thesis does not focus on IPv6 based communication or application layer protocols such as HTTP and COAP, the custom network layer connects with underlying IEEE 802.15.4 TSCH-MAC layer and radio layer only.

5.5.1 Data Structures and Functions

Contiki-NG implements network driver with certain data structures and functions. The OS uses a packet buffer that is used to create a outbound packet or store a inbound packet. The buffer is also used to operate on a packet and only one packet of either outbound packet or inbound packet can be stored in the buffer at a time. Each network driver has a name , an function to initialise the network stack, an input function that is triggered by MAC layer to notify about an incoming packet that is stored in packetbuffer and an output function to trigger the MAC to send the outgoing packet stored in packetbuffer.

```
struct network_driver {
  char *name;

  /** Initialize the network driver */
  void (*init)(void);

  /** To get notified of incoming packet in packetbuffer. */
  void (*input)(void);

  /** Output function, stores the packet in packetbuffer and notifies the
    MAC Layer to about outgoing packet . */
  uint8_t (*output)( void *from , uint16_t len, const linkaddr_t *dest);
};

const struct network_driver neuralnet_driver = {
  "neuralnet",
  init,
  input,
  neural_send
};
```

Listing 5.1: Structure of custom network driver in Contiki-NG

The input function has an input sniffer which is called just before the packet goes from the MAC layer up the custom network layer. The input callback function defines routines that operate on the incoming packet. The output function (neural_send) packs the outgoing data into packet buffer with destination address and notifies the MAC layer.

```

struct netstack_sniffer {
  struct netstack_sniffer *next;
  void (*input_callback)(void);
  void (*output_callback)(int mac_status);
};

#define NETSTACK_SNIFFER(name, input_callback, output_callback) \
  static struct netstack_sniffer name = { NULL, input_callback,
   output_callback }

void netstack_sniffer_add(struct netstack_sniffer *s);
void netstack_sniffer_remove(struct netstack_sniffer *s);

static void input(void)
{
  if(callback)
      callback->input_callback();
}

uint8_t neural_send( void *from , uint16_t len, const linkaddr_t *dest)
{
  neuralnet_buf = from;
  neuralnet_len = len;
  packetbuf_clear();
  packetbuf_copyfrom(neuralnet_buf, neuralnet_len);

  if(dest != NULL) {
    packetbuf_set_addr(PACKETBUF_ADDR_RECEIVER, dest);
  } else {
    packetbuf_set_addr(PACKETBUF_ADDR_RECEIVER, &linkaddr_null);
  }
  packetbuf_set_addr(PACKETBUF_ADDR_SENDER, &linkaddr_node_addr);
  LOG_INFO("sending %u bytes to ", packetbuf_datalen());
  LOG_INFO_LLADDR(packetbuf_addr(PACKETBUF_ADDR_RECEIVER));
  LOG_INFO("\n");
  NETSTACK_MAC.send(NULL, NULL);
  return 1;
}
```

Listing 5.2: Structure of sniffer, input and send functions of custom network layer in Contiki-NG

The memory occupied by custom network stack shown in Table. 5.2 is below 16 KB and fits into the Zolertia firefly node.

.text (bytes)	.data (bytes)	.bss (bytes)	dec (bytes)
35742	445	13584	49771

Table 5.2: Minimum static memory occupied by custom network stack on Contiki-NG OS

5.6 Neural Network Optimisation

Since the input layer of a NN depends on the nodes deployed on the ceiling and 23 nodes are used in this thesis for experimentation, the input layer size is 23 for central localisation method and 44 for distributed localisation method. The output layer size is 2 for both method. By varying the hidden layer size, we can achieve the optimal network in terms of size that fits into the Zolertia firefly node as shown in Table. 5.3. The data buffer, weights and biases use float values which are assumed to be 4 bytes and the available memory(E_{av} according to Table. 5.2 is 2355 bytes.

$L_{databuffer}$	$N_{neurons}$	Hidden neurons	No.of Weights ($N_{weights}$)	E_t (bytes)	E_{av} (bytes)
23	31	4	100	1232	2355
23	32	5	125	1440	2355
23	33	6	150	1648	2355
23	34	7	175	1856	2355
23	35	8	200	2054	2355
44	28	1	46	1112	2355
44	28	2	92	1488	2355
44	28	3	138	1864	2355
44	28	4	184	2240	2355

Table 5.3: Minimum static memory occupied by custom network stack and NN system on Contiki-NG OS

5.7 Experimental set up

5.7.1 Gravel net

Gravel net is a experimentation and testing facility for IoT hardware, protocols and applications [2].

5.7.2 Fingerprinting phase

The first phase in the experimental process to be accomplished is the offline phase. This involved building up of required database for fingerprinting method. A set of 5000 RSSI samples for each of the 31 training positions were taken by keeping the target device at these positions. Each of the 23 anchor nodes acquired 5000 samples and sent the data to the gravel network [2]. The data was downloaded from the gravel server and a database was created on a computer.

5.7.3 Training Phase

5.7.3.1 Simulation Platform

The next stage in the offline phase is to train the neural network model parameters with the generated databases. The parameters were trained on a **DELL Inspiron 15 7000** laptop with Ubuntu 18.04 as Linux operating system. Simulation experiments of the network were performed for different learning rates, training samples and changing the number of hidden neurons. The training was performed for 600 epochs. According to the Fig:5.8,

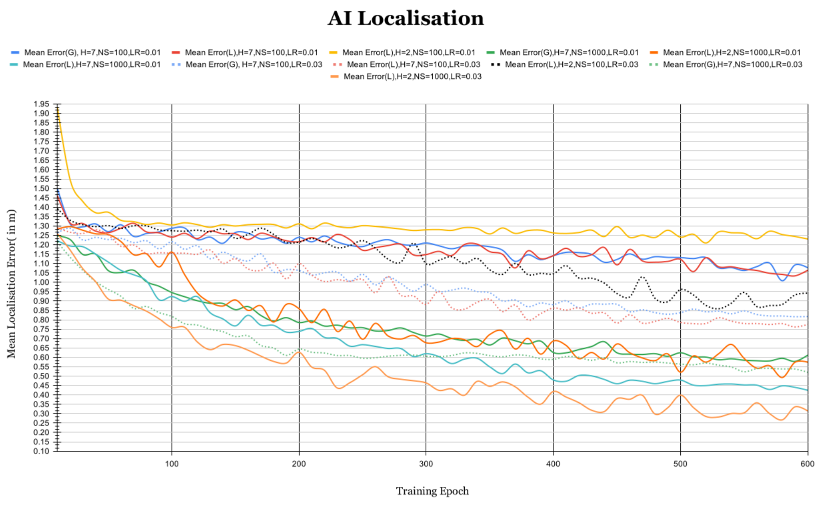

Figure 5.8: Localisation error for central and distributed method

the local prediction algorithm with H=2, NS=100 and LR=0.01, where H is number of neurons in hidden layer, NS is number of training samples, and LR is the learning rate of the NN shows the worst performance with an error of **≈1.30m**. For central method, the simulation for H=7, NS=100 and LR=0.01 shows the worst performance with an error of **≈1.25m**. The best performances were shown for H=7, NS=1000 and LR=0.03 for central mehtod with an error of **≈0.65m** and H=2, NS=1000, LR=0.03 for distributed method with an error of **≈0.35m**. The model parameters(i.e. the weights and biases) of these two best working network models were deployed in the real-time network in the online phase.

5.7.4 Estimation phase

The final phase in the experimental process is the online phase. This involved flashing the anchor nodes with Contiki-NG operating system along with chosen network model parameters selected in the simulation phase and performing the localisation of the target device using the two methods.

Central localisation method

The output node has the highest level L_l in the network which takes RSSI measurement from itself and all other nodes present at level L_n (where n=1, 2, 3, ..., l-1) propagate data towards the output node. The data collection can be divided into three steps:

1. Initializing:
   - Target device (TD) sends out broadcast messages.
   - Upon receiving broadcast messages, nodes record TD's RSSI value in local memory. In order to avoid flooding the data buffer of output buffer with values measured by itself, only the first broadcast message is stored in data buffer. The subsequent values are ignored until a cycle of prediction is finished.
2. Communication:
   - Nodes at level L_n pick up the node whose level is L_{n+1} in the list and send messages containing RSSI value received in step 1 to the node.
   - Nodes also forward received messages from neighbors present at L_{n-1} to nodes present in higher level L_{n+1} from the list.
3. Data collection:
   - Step 1 and 2 are repeated as long as the TD is active.
   - The output node stores the RSSI values in the data buffer whose capacity is pre determined depending on number of anchor nodes in the network. Once the buffer capacity is full, the data is sent to feed-forward neural algorithm for location prediction. After each prediction, the buffer is erased and filled again with new values.

Distributed localisation method

The values predicted by the node which forms the network are considered as final output. This node has the highest level L_l and all other nodes are present at level L_n (where n=2, 3, ..., l-1) and L_1. The data collection of each node can be divided into three steps:
For nodes at level L_1 :

1. Initializing:
   - Target device (TD) sends out broadcast messages.
   - Upon receiving broadcast messages, nodes record TD's RSSI value in local memory. In order to avoid flooding the data buffer of each node with values measured by itself, only the first broadcast message is stored in data buffer. The subsequent values are ignored until a cycle of prediction is finished.

2. Communication:

   - Each node picks up the node at L_2 in its list and sends messages containing the recorded RSSI value received in step 1 to the L_2 node.

3. Data collection:

   - Step 1 and 2 are repeated as long as the TD is active.
   - Nodes receive messages of RSS values from the nodes at level L_2 and stores in the data buffer whose capacity is equal to the size of input layer of NN. Once the buffer capacity is full, the data is sent to feed-forward neural algorithm for location prediction which is later propagated into the network via their neighbor L_2 node. After each prediction, the buffer is erased and filled again new values.

For nodes at level L_n :

1. Initializing:

   - Target device (TD) sends out broadcast messages.
   - Upon receiving broadcast messages, nodes record TD's RSSI value in local memory. In order to avoid flooding the data buffer (b_1) of each node with values measured by itself, only the first broadcast message is stored in data buffer. The subsequent values are ignored until a cycle of prediction is finished.

2. Communication:

   - Each node sends messages containing the recorded RSSI value received in step 1 to all its one hop neighbors.
   - Each node forwards received messages and also sends its output values to its one hop neighbors. If a node is at level L_2, it only forwards RSSI values to nodes at L_1 from its list.

3. Data collection:

   - Step 1 and 2 are repeated as long as the TD is active.
   - Each node receives messages of RSS values from nodes at level L_n and store in b_1. The messages which contain output values of other nodes are stored in another buffer b_2. The sum of capacities of b_1 and b_2 is equal to te size of input layer of NN. When both the buffer capacities are full, the data is sent to feed-forward neural algorithm for location prediction and later the (x,y) values (except for the node whose output is considered as final output) propagated into the network via their one hop neighbors. After each prediction, the buffers are erased and filled again with new values.

5.8 Evaluation metrics

5.8.1 Positioning accuracy

The first evaluation standard of positioning technology is positioning accuracy. The accuracy of algorithm is influenced by reflection/refraction, multi-path fading, irregular building geometry, noise of electrical equipments in the lab as well as obstacles of wall.

5.8.2 Rate of Predictions

In order to understand the throughput of the network, the prediction of location of the target device is used. This helps to understand the latency of network and the duration of time needed to place the target device at a position in order to predict its location.

5.8.3 Packet Reception Ratio(PRR)

This metric is measured by the output neural node. Two separate measurements are made. First measurement involves PRR with neighbour nodes and second measurement involves with the target device.

Chapter 6

Results

This section explains the results that are derived from the execution of global and local prediction algorithms. It shows the obtained metrics and how they vary for both algorithms. The results are obtained once the stable network is formed and values are taken after placing the target device at each location after 5 minutes.

6.0.1 Localisation

Central method

In this method, the system was tested by placing target user at training and test locations as shown in Fig: 5.3. In real system, the prediction was better than in simulation at locations where the density of training points were more and there were less obstacles. The average error at these points was between the range of 0.5m to 1m. However, the error increased at points where there were more obstacles and if there were no or few training points.

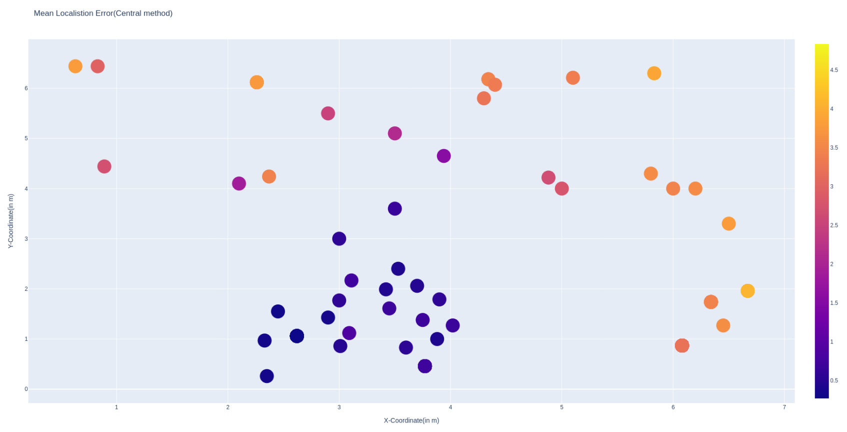

Figure 6.1: Mean localisation error for central method

Distributed method

In this method, the system was tested by placing target user at training and test locations as shown in Fig: 5.3. In real system, the average error at locations where the density of training points were more and there were less obstacles was between the range of 0.5m to 1m. However, the error increased at points where there were more obstacles and if there were no or few training points.

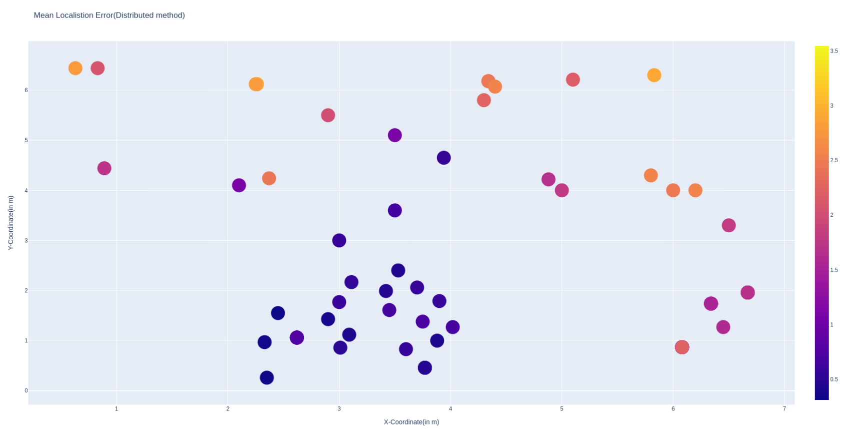

Figure 6.2: Mean localisation error for distributed method

Discussion

Both the prediction techniques performed better at positions where density of test points were high. However, at sparse points the distributed method performed better than central method with maximum observed mean error was in the range of 3m to 3.5m and in central method, it was between 4m to 4.5m. This is because in distributed method, there are more inputs to each node and also each node predicts the location of the target user.

6.0.2 Prediction Rate

The target user broadcasts packets at the rate of 1 packet per second. After testing the network for 25 minutes, it was observed that central method prediction throughput was 29 every minute which is higher than distributed method prediction throughput which was 11 every minute

Discussion

The throughput in central method is higher because the input size is 23 which is less compared to distributed method. Also, the data from each anchor node is directed towards the sink (i.e. output node) forming a unidirectional path and hence the network traffic is less. In distributed method, the input size of leaf nodes is 23 and their prediction is faster.

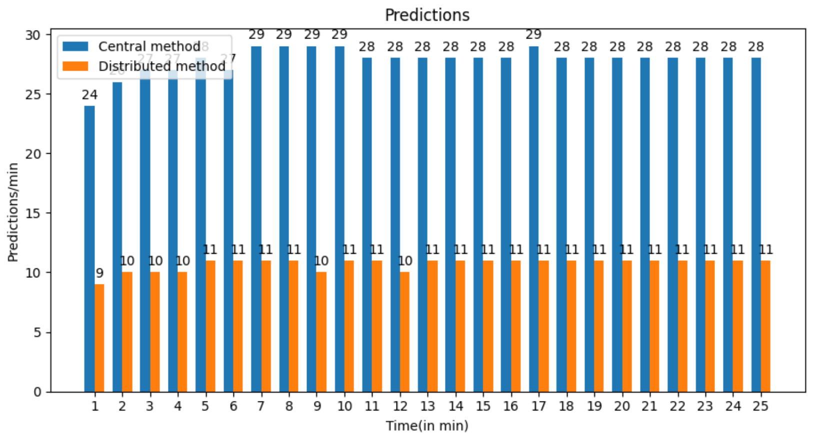

Figure 6.3: Rate of predictions with time with 1packet/s of target user

However, other nodes have an input size of 67 and have two input buffers, one for RSSI values (23) and other for coordinate values (44). The prediction happens only when both the buffers are full. Due to these factors the prediction rate is lower.

6.0.3 Packet reception ratio

The PRR metric was measured by the output node with its neighbors and the target user. The PRR at the output node was observed by varying the number of nodes that formed the network. For each variation, a new network was formed. The results showed that strong links were formed between output node and its neighboring nodes with PRR of around 96% in every variation.

Discussion

The target user broadcasts packets to the network. Its PRR is lower as the timeslot is shared by all the nodes in network and also used to transmit periodic enhanced beacons which causes contention. The PRR of neighbors is high as there are separate timeslots for each neighbor and slots are increased if the traffic load increases which results in reduction in interference and packet loss. The higher PRR is in accordance with TREE [27] algorithm.

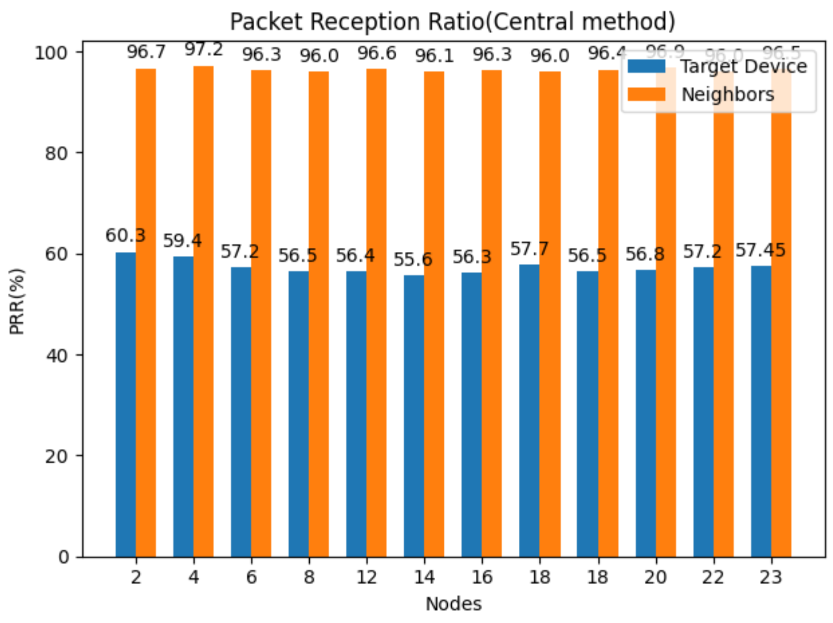

Figure 6.4: Packet Reception Ratio in central method

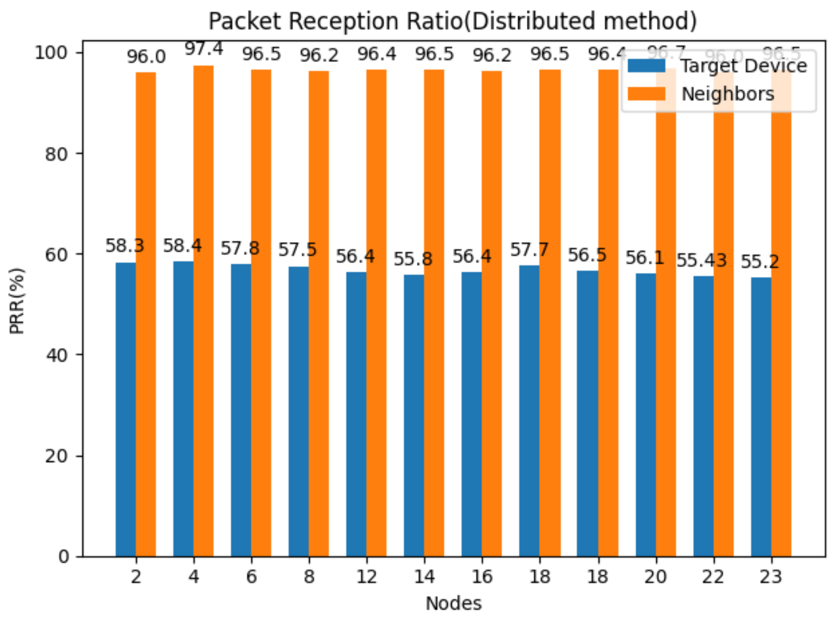

Figure 6.5: Packet Reception Ratio in distributed method

Chapter 7

Conclusions and future work

7.1 Conclusions

The main goal of this master thesis was to design and implement AI based localisation methods that uses fingerprinting approach and optimised ANN models deployed on an IEEE 802.15.4 based network of low power devices that use harvested energy.

Two ANNs with input sizes 23 and 67 were designed for central and distributed method respectively and successfully deployed using Zolertia Firefly on Contiki-NG OS with a custom network layer based on IEEE 802.15.4 standard. Implementation of system was evaluated by performing fingerprinting based real time indoor localisation in distributed and central computing method. The system provided an average localisation accuracy of under 1m which is comparable to performances in other paper and a throughput of 29 predictions (central method) and 11 predictions (distributed method) per minute.

7.2 Future Work

The following is a list of possible future topics:

- In the current system, the network is formed initially and it's assumed that all the nodes will be active during the lifetime of network. However, certain nodes can be inactive and needs to be monitoring. Improvements by using algorithms that manage the network in real time could be done.

- All the links in the network have the same weight. This could be changed and different weights could be added to link depending on the error these links produce in real network.

- In this architecture, floating point operations are used by prediction algorithms that require sizeable amount of memory and restricts size of NN. Binary neural networks (BNN) that use binary weights could be implemented in future as they require very less memory. By reducing training set and using BNN, networks can be trained in real time directly on the nodes.

Bibliography

[1] A open source project, “FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions”. https://www.freertos.org/.[Accessed: 11-Jan-2021]. 12

[2] ”gravel/net”. https://gravel.network/. 29, 36

[3] Zolertia: Firefly. [online]Available: http://zolertia.io/product/hardware/firefly. 28, 29

[4] “Contiki-NG: The next generation contiki, the open source os for the internet of things”. https://contiki-ng.org/.[Accessed: 09-Jan-2021]. 12

[5] “Contiki: The official git repository for contiki, the open source os for the internet of things”. 12

[6] “RIOT - the friendly operating system for the internet of things”. https://riot-os.org.[Accessed: 11-Jan-2021]. 12

[7] S Andrew. Tanenbaum–computer networks –prentice hall. *New Jersey*, 2003. ix, 7

[8] NXP application note JN-UG-3024. Ieee 802.15.4 stack user guide. 2 2014. ix, ix, ix, 10, 11

[9] Abd Elgwad M. El Ashry and Bassem I. Sheta. Wi-fi based indoor localization using trilateration and fingerprinting methods. *IOP Conference Series: Materials Science and Engineering*, 610:012072, oct 2019. 17

[10] E. Basar, M. Di Renzo, J. De Rosny, M. Debbah, M. Alouini, and R. Zhang. Wireless communications through reconfigurable intelligent surfaces. *IEEE Access*, 7:116753–116773, 2019. 1

[11] Lukas Cavigelli and Luca Benini. Cbinfer: Exploiting frame-to-frame locality for faster convolutional network inference on video streams. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(5):1451–1465, 2019. 26

[12] Agnieszka Choroszucho and Bogusław Butryło. Local attenuation of electromagnetic field generated by wireless communication system inside the building. *Przeglad Elektrotechniczny*, 87(7):123–126, 2011. 1

[13] André Cunha, Anis Koubaa, Ricardo Severino, and Mário Alves. Open-zb: an open-source implementation of the ieee 802.15.4/zigbee protocol stack on tinyos. pages 1–12, 10 2007. ix, 8

[14] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010. ix, 28

[15] Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Steven Bohez, Pieter Simoens, Piet Demeester, and Bart Dhoedt. Distributed neural networks for internet of things: The big-little approach. In *International Internet of Things Summit*, pages 484–492. Springer, 2015. ix, 16, 18, 26

[16] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. Orchestra: Robust mesh networks through autonomously scheduled tsch. 11 2015. 32

[17] Simon Duquennoy, Atis Elsts, Beshr Al Nahas, and George Oikonomou. Tsch and 6tisch for contiki: Challenges, design and evaluation. 06 2017. ix, 32

[18] Umar Farooq, Muhammad Amar, KM Hasan, M Khalil Akhtar, Muhammad Usman Asad, and Asim Iqbal. A low cost microcontroller implementation of neural network based hurdle avoidance controller for a car-like robot. In *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, volume 1, pages 592–597. IEEE, 2010. 15, 18

[19] GMDT Forecast. Cisco visual networking index: global mobile data traffic forecast update, 2017–2022. *Update*, 2017:2022, 2019. 1

[20] C. Huang, G. C. Alexandropoulos, C. Yuen, and M. Debbah. Indoor signal focusing with deep learning designed reconfigurable intelligent surfaces. In *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, 2019. 1

[21] Texas Instruments. Cc2538 powerful wireless microcontroller system-on-chip for 2.4-ghz ieee 802.15. 4, 6lowpan, and zigbee applications. *CC2538 datasheet (April 2015)*, 2015. ix, 29

[22] Rukaiya Javaid, Rehan Qureshi, and Rabia Enam. Rssi based node localization using trilateration in wireless sensor network. *Bahria University Journal of Information Communication Technologies*, 8:58–64, 12 2015. 17

[23] Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, and Sarah Mack. *Principles of neural science*, volume 4. McGraw-hill New York, 2000. 5

[24] Ali Khalajmehrabadi, Nikolaos Gatsis, and David Akopian. Modern wlan fingerprinting indoor positioning methods and deployment challenges. *IEEE Communications Surveys & Tutorials*, 19(3):1974–2002, 2017. 3

[25] B Komal. Process, Contiki OS. https://komalbattula.wordpress.com/process-scheduling/. [Accessed: 01- Feb- 2021]. ix, 14

[26] Anis Koubâa, Mário Alves, and Eduardo Tovar. Ieee 802.15. 4 for wireless sensor networks: a technical overview. 2005. 8

[27] Tim Van Der Lee, Georgios Exarchakos, and Sonia Heemstra De Groot. Distributed reliable and energy-efficient scheduling for lr-wpans. *ACM Transactions on Sensor Networks (TOSN)*, 16(4):1–20, 2020. 32, 43

[28] Michele Magno, Michael Pritz, Philipp Mayer, and Luca Benini. Deepemote: Towards multi-layer neural networks in a low power wearable multi-sensors bracelet. In *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 32–37. IEEE, 2017. 26

[29] Steffen Nissen et al. Implementation of a fast artificial neural network library (fann). *Report, Department of Computer Science University of Copenhagen (DIKU)*, 31:29, 2003. 26

[30] F. Oldewurtel and P. Mahonen. Neural wireless sensor networks. In *2006 International Conference on Systems and Networks Communications (ICSNC'06)*, pages 28–28, 2006. ix, 5, 6

[31] Antonio Pullini, Davide Rossi, Igor Loi, Giuseppe Tagliavini, and Luca Benini. Mr. wolf: An energy-precision scalable parallel ultra low power soc for iot edge processing. *IEEE Journal of Solid-State Circuits*, 54(7):1970–1981, 2019. 26

[32] Raul Rojas. The backpropagation algorithm. pages 149–182, 1996. 22

[33] S. Sadowski and P. Spachos. Rssi-based indoor localization with the internet of things. *IEEE Access*, 6:30149–30161, 2018. 2

[34] L Saad Saoud and Abdelhafid Khellaf. A neural network based on an inexpensive eight-bit microcontroller. *Neural Computing and Applications*, 20(3):329–334, 2011. 15, 18

[35] Erhan Sesli and Gökçe Hacıoğlu. Contiki os usage in wireless sensor networks (wsns). *Turkish Journal of Electromechanics & Energy*, 2(2):1–6, 2017. ix, 31

[36] Ali Shareef, Yifeng Zhu, and Mohamad Musavi. Localization using neural networks in wireless sensor networks. In *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, pages 1–7. Citeseer, 2008. 3, 18

[37] Shanpu Shen, Yujie Zhang, Chi-Yuk Chiu, and Ross Murch. A triple-band high-gain multibeam ambient rf energy harvesting system utilizing hybrid combining. *IEEE Transactions on Industrial Electronics*, 67(11):9215–9226, 2019. 3

[38] Rashmi Sharan Sinha and Seung-Hoon Hwang. Improved rssi-based data augmentation technique for fingerprint indoor localisation. *Electronics*, 9(5), 2020. 17

[39] Santosh Subedi and Jae-Young Pyun. Practical fingerprinting localization for indoor positioning system by using beacons. *Journal of Sensors*, 2017:1–16, 12 2017. 17

[40] J. Tang, D. Sun, S. Liu, and J. Gaudiot. Enabling deep learning on iot devices. *Computer*, 50(10):92–96, 2017. 3

[41] J. Tang, D. Sun, S. Liu, and J. Gaudiot. Enabling deep learning on iot devices. *Computer*, 50(10):92–96, 2017. 15

[42] Jin Wang, Yu Gao, Wei Liu, Arun Kumar Sangaiah, and Hye-Jin Kim. An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 15(3):1550147719839581, 2019. 15, 18

[43] Wenxu Wang, Damián Marelli, and Minyue Fu. Fingerprinting-based indoor localization using interpolated preprocessed csi phases and bayesian tracking. *Sensors*, 20:2854, 05 2020. 17

[44] X. Wang, M. Magno, L. Cavigelli, and L. Benini. Fann-on-mcu: An open-source toolkit for energy-efficient neural network inference at the edge of the internet of things. *IEEE Internet of Things Journal*, 7(5):4403–4417, 2020. 26

[45] Tim Winter, Pascal Thubert, Anders Brandt, Jonathan W Hui, Richard Kelsey, Philip Levis, Kris Pister, Rene Struik, Jean-Philippe Vasseur, Roger K Alexander, et al. Rpl: Ipv6 routing protocol for low-power and lossy networks. *rfc*, 6550:1–157, 2012. 33

[46] L. Yu, M. Laaraiedh, S. Avrillon, and B. Uguen. Fingerprinting localization based on neural networks and ultra-wideband signals. In *2011 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 184–189, 2011. 2

[47] F. Zafari, A. Gkelias, and K. K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials*, 21(3):2568–2599, 2019. 2