Expertise
and insight
for the future

Joni Nurmimetsä

Analysis of Current IoT Mesh Technologies and Sample Implementation for Automotive Application

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

21 April 2020

Author	Joni Nurmimetsä
Title	Analysis of current IoT mesh technologies and sample implementation for automotive application
Number of Pages	50 pages
Date	21.4.2020
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Instructors	Keijo Länsikunnas, Senior Lecturer

This thesis analyses and catalogues current day Internet of Things (IoT) protocols in relation to their relevancy to different mesh networking technologies. A large number of protocols are today proposed for different specific applications in the IoT field, but only a few of them have mesh networking integrated as an inherent feature of the respective protocol stacks. The aim of this thesis is to provide understanding of underlying technologies and narrow down the wide field of currently available main technologies.

This thesis reviews openly available documentation and other online material on different protocols and describes their relative strengths and weaknesses. This review also shows how different protocols and their major design choices have affected the popularity and vitality of their respective development ecosystems.

The findings are that, while there certainly exists a large number of more or less specialised protocols utilised in the field, the development and support investments have made only a few protocols which are relatively widely adopted.

In conclusion, this thesis demonstrates and presents the benefits offered by a lightweight mesh connected sensor network in an automotive application. In successfully implementing this, it is shown that the benefits are real and easily obtainable.

Keywords	Internet of Things, mesh network, Bluetooth LE, Thread, Z-Wave, ZigBee

Tekijä Otsikko Sivumäärä Aika	Joni Nurmimetsä Esineiden internetin solmuverkkoteknologiat ja esimerkkitoteutus käytännön sovelluksesta ajoneuvokäytössä 50 sivua 21.4.2020
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tietotekniikka
Ohjaaja	Lehtori Keijo Länsikunnas

Insinöörityön tarkoituksena oli analysoida ja koota tämän hetken esineiden internetin protokollia suhteessa niiden merkityksellisyyteen eri solmuverkkoteknologioiden suhteen. Erilaisiin sovelluksiin on nykyään käytettävissä suuri määrä IoT-protokollia, mutta vain muutamissa niistä solmuverkot ovat lähtökohtaisesti mukana ominaisuutena. Insinöörityön tarkoituksena oli koota katsaus käytettyihin teknologioihin ja keskittyä muutamaan tällä hetkellä saatavilla olevaan pääteknologiaan.

Insinöörityössä perehdyttiin avoimesti saatavilla oleviin dokumentteihin ja muihin verkkomateriaaleihin eri protokollista ja pyrittiin kuvaamaan niiden suhteellisia vahvuuksia ja heikkouksia. Insinöörityöstä selvitettiin, miten eri protokollat ja niiden suunnittelulähtökohdat ovat vaikuttaneet näiden protokollien ja niiden ekosysteemien suosioon sekä elinvoimaisuuteen. Useat tutkituista protokollista ovat painottaneet yhteensopivuutta helpon laajennettavuuden ja protokollan avoimuuden kustannuksella, mikä heijastuu usealla eri tavalla sekä sovelluskehittäjille että loppukäyttäjille.

Loppupäätelmä oli, että vaikka nykyisin tarjolla on suuri määrä enemmän tai vähemmän erikoistuneita protokollia, vain muutamat niistä ovat laajalti käytössä, mikä on seurausta tietoisista panostuksista kehitys- ja tukitoimintoihin.

Insinöörityön osana tehtiin esimerkkisovellus käyttäen yhtä analysoiduista protokollista ja kaupallisesti saatavilla olevaa kehitysalustaa. Insinöörityöraportissa käytiin läpi sovelluksen eri osa-alueet laitteiston, ohjelmiston, kehitysympäristön sekä käytetyn pilvialustan osalta. Esimerkkisovelluksen toteutuksen voidaan todeta vahvistaneen aiemmin kuvattujen solmuverkkoteknologioiden etujen saavutettavuus ajoneuvokäytössä. Toimivan implementaation rakentaminen todisti näiden etujen olevan todellisia ja helposti saavutettavissa.

Avainsanat	Internet of Things, mesh network, solmuverkot, esineiden internet

Contents

List of Abbreviations

3GPP	3rd Generation Partnership Project. Cellular technology industry standards body.
API	Application Programming Interface. A set of defined functions and procedures exposing an application or service.
DSRC	Dedicated Short-Range Communications. Short-range to medium-range wireless communication technology designed for automotive use.
EDGE	Enhanced Data rates for GSM Evolution. Improved technology for allowing higher data speeds in pre-3G mobile networks.
HWMP	Hybrid Wireless Mesh Protocol. A routing protocol used by the 802.11s standard.
IDE	Integrated Development Environment. An application providing a comprehensive set of functionalities for software development.
IoT	Internet of Things. An overarching set of technologies allowing low computing resource devices and objects to connect to networks and transmit data.
IrDA	Infrared Data Association. An industry interest group that has defined an optical wireless protocol of the same name using infrared light.
M2M	Machine to Machine. A term describing machine-to-machine communications.
NB	Narrow Band. A term describing limited communications bandwidth.
NFC	Near-Field Communication. Very close proximity communications technology.
OSI	Open Systems Interconnection. A system describing concepts and models for telecommunications and computing systems, developed by ISO.
OTA	Over the Air. A concept describing wireless communications to reprogram networked mobile devices.
PDU	Protocol Data Unit. A single unit of information transmitted in a data network.
RFC	Request for Comments. Text documents describing internet related technologies, some of which have later became the foundational standards for the Internet.

RFID	Radio-Frequency Identifier. A term describing devices allowing wireless identification of objects or devices.
RoW	Rest of the World. In telecoms, typically non-North American markets.
SAR	Segmentation and Reassembly. Action where a device has to divide, and on the other end, reassemble, a single PDU into multiple packets due to maximum transmit unit limitations.
TTL	Time to Live. Maximum transmission or lifetime limit for data being stored or transmitted.
V2X	Vehicle-to-X. Term describing several possible scenarios where a vehicle can transmit data to multiple other entities such as other vehicles (V), Infrastructure (I) or any other receiver.
WAN	Wide Area Network. A network covering a wide geographical area.
WLAN	Wireless Local Area Network. A wireless network covering a smaller area, typically a single building or locale.
WPAN	Wireless Personal Area Network. A wireless network covering an area of a single person.

1 Introduction

This thesis analyses the current state of different IoT mesh networking technologies. The IoT field has been quickly maturing and first generations of deployments are now being phased out. Mesh networking as an idea has a long history, the Internet itself being one, but it has not been extensively applied to the IoT field previously.

The first generation of what today would be considered IoT technologies were called M2M or machine-to-machine technologies and were based on cellular technologies, mainly 2G/2.5G(EDGE) networks. These deployments were quickly proven to have insufficient spectral efficiency and to be too expensive to support the envisioned number of nodes for future IoT deployments. Some industry suppliers suggested that the number of IoT nodes might reach tens of billions by 2020, which is several multiples higher than the number of mobile devices globally today and was certainly several orders of magnitude more than the number of M2M/IoT nodes in existence when the prediction was made. [1.]

To counter these challenges, two main approaches have emerged in the IoT field in general. On the one hand, to lower cost and to make devices more spectrally efficient and energy efficient, several narrow band (NB) radio network technologies have been developed. While these are not mesh networks by definition, they are shortly discussed to give readers a broader overall picture. Another reason to discuss them is that a few edge cases exist. The second approach has been to create technologies to implement short range dense node-to-node networks where the main driver for development has been low cost and low power consumption. In addition, one of the aims has been to require close to zero configuration while remaining fault tolerant and providing a reliable communication channel through auto configuring node-to-node communications topology. These technologies constitute what are commonly referred to as mesh networks. [2.]

This thesis concentrates on providing a current day picture of the IoT mesh networking technologies, including a brief historical background as well as analysis of several of the major protocols in the field today.

2 General overview of applicable communication technologies

2.1 Comparison of (W)WAN, (W)LAN and (W)PAN

There are certainly a wide number of technologies that are relevant to the IoT communications field, but not all of them are relevant when discussing the current state of mesh networking technologies. In the context of IoT mesh networks, the major division is between different WAN, (W)LAN and (W)PAN technologies. While these refer to distinctly different networking technologies and define separate technical domains, “mesh networks” can be used to refer to products in any of the aforementioned categories. In technical press, these fields are frequently conflated especially when discussing IoT products and services. This section shortly explores what is considered relevant in the context of this thesis.

Additionally, it should be noted that only wireless radio frequency technologies are considered in this thesis. While there are certainly protocols and technologies that allow for wired mesh networks, most notably Ethernet and KNX based networks, these are not considered to be relevant in the IoT mesh networking context. Some IoT technology standards also consider and define specifications for short-range protocols such as Infrared Data (IrDA), Near-Field Communication (NFC) and Radio-Frequency Identifier (RFID). While these are relevant to IoT products, they are not relevant to mesh networks as underlying protocols. In addition, certain highly specialised magnetic induction-based technologies are discarded, even though they are used for sensor networks in applications that could be considered to be the core the IoT field.

2.2 Relevant (W)WAN technologies

Wireless wide area radio networks are and have been one of the core enabling technologies for all IoT deployments. More specifically previous large scale M2M or IoT deployments have used 2G or 2.5G EDGE networks for connectivity, the most often cited example being several national deployments of “smart”, i.e. connected electricity meters. While technology has moved forward and today 5G networks are being deployed, the focus has been in increasing network throughput, supporting a higher number of subscribers and lowering network latency. None of these parameters is directly relevant

for many IoT applications, where especially throughput and latency are already sufficient today to meet the requirements. Primary drivers in IoT application development are low cost and low power usage. To fulfil these requirements for IoT applications, three different protocols and protocol families have emerged. [3.]

2.2.1 3GPP defined standards

3GPP is the standards organisation behind the current generation of mobile networking technologies. As such their IoT efforts build on the existing infrastructure and specifications. LTE-M, NB-IoT and EC-GSM-IOT, which are the three major protocols defined for different use cases, are all based in using star topology and are not relevant to this thesis, apart from being one of the applicable access methods for providing connectivity for IoT mesh network access or gateway nodes. [4.]

It should also be noted that since 3GPP represents the current interests and vision of the existing operator and technology landscape, the standards preclude any possibility of alternative approaches into (W)WAN networking. All of the above-mentioned protocols are geared towards ensuring that the current model of deployment is kept unchanged.

2.2.2 Sigfox

Sigfox is a proprietary networking technology developed by a French company of the same name. The protocol exchanges data transmission rate and latency for very low power requirements and good signal coverage. As a consequence, the message and data rates are extremely low and Sigfox nodes can only transmit up to 140 messages with 8 bytes of data, per day. The network architecture is also exclusively based on star topology, which is required to keep the node hardware very simple and cheap to manufacture. Due to these limitations, it is not suitable to be used even as a mesh network gateway connectivity. [5.]

2.2.3 LoRaWAN

The LoRaWan protocol differs from the 3GPP family of protocols and from Sigfox in that it is intended as an open standards approach to providing low cost and low power

network technology for IoT applications. However, it builds on proprietary LoRa radio technology developed by the semiconductor manufacturer Semtech [6.].

The LoRaWan protocol family also has similar limitations as Sigfox with very low bandwidth. Similarly, the protocol specifications limit the network to star topology and there are no provisions to creating mesh networks [7.].

It should be mentioned that there are IoT mesh network development platforms available (PyCom LoPy4 / FiPy) that have LoRa hardware integrated, but even they do not envision it to be used for creating mesh networks. These platforms utilise other wireless protocols such as Bluetooth, which are similarly integrated into the same hardware, to form the mesh networks.

2.2.4 DASH7

DASH7 originated from American military communications technology development in 2009. It was derived from the ISO18000-7 standard for RFID technology but has lost compatibility with the original specification along with further development. DASH7 Alliance was formed to promote and ensure interoperability of the standard, which has continued to be developed by its members. The latest published specification is DASH7 Alliance Protocol v1.2 released in January 2019. [8.]

The focus of DASH7 was to provide a very low footprint and low power consumption protocol for asset tracking purposes while still enabling node-to-node communications. To this end, it exclusively uses sub-1GHz frequency bands and is based on the premise that nodes will only infrequently broadcast messages. This also implies basic star shaped network topology and the protocol does not include any provisions for nodes retransmitting messages, thus excluding mesh networking as an option. As such this protocol will not be further discussed in this thesis. [9.]

2.3 Relevant (W)LAN technologies

2.3.1 802.11 Wi-Fi

When IoT technologies and any connectivity solutions are discussed today, it is natural that the currently deployed IEEE 802.11 based Wi-Fi technologies is inherently included as one of the foundational technologies that have enabled ubiquitous connectivity underlying the entire IoT field. One of the more recent additions to scale and improve Wi-Fi networks have been the addition of mesh networking. Several widely available and well-known Wi-Fi networking products, such as Google Wifi, NETGEAR Orbi or Ubiquity UniFi, have made mesh networking one of their core features and the main selling points. [10.]

This is also the specific context in which these Wi-Fi mesh networks might be most easily confused being relevant to IoT mesh networks. While Wi-Fi is certainly widely used as a pure access medium for different kinds of IoT nodes and terminals, it does not include provisions for allowing network clients to retransmit data to extend network reach, which is one of the defining features for mesh networks. Mesh networking is simply used to provide alternate backhaul data and a control plane for the base stations to communicate between themselves, thus increasing network coverage. Individual Wi-Fi network clients, be they IoT devices or not, do not form part of or connect into the mesh and are simply utilising the available network service as if it was any other standard Wi-Fi network.

However, due to the extendable nature of IEEE standards there are some very specific sub-specifications under the 802.11 family that do include mesh networking elements extended all the way to end nodes. These are shortly discussed in two following subsections.

2.3.2 802.11p / DSRC / ITS-5G

IEEE 802.11p was originally published in July 2010 and was created as a protocol to enable communication in vehicular environments, for both vehicle-to-vehicle and for vehicle-to-infrastructure applications. Protocols based on this specification were developed in both in the United States and in Europe. In the United States, it became a DSRC specification published by the Department of Transportation, but no official rules

were ever published mandating actual use. In Europe, the ITS-5G specification was published as part of the Intelligent Transportation System platform, also making any implementation optional. It should also be noted that ITS-5G should not be confused with mobile 5G technologies as there is no direct relationship, other than roughly parallel technological fields. [11.]

However, even after almost 10 years, neither DSRC nor ITS-5G have seen any significant deployments in actual production or end product environments. Today it seems that the manufacturers are seeing future technological development in different mobile networking technologies such as LTE or 5G mobile networks, rather than in a dedicated automotive V2X technology. Neither protocol is further discussed in this thesis due to lack of actual real-life implementation examples outside testing or lab environments. [12.]

2.3.3 802.11s

802.11s was designed as the mesh networking extension for standard 802.11 Wi-Fi networks. In 2012 when it was standardised and included in the main 802.11 specification, the envisioned usage was to enable traditional Wi-Fi networks to utilise wireless medium to extend their footprint, which is one of the key benefits of features of any mesh technology. While specification allows for any Mesh Station (Mesh STA) with compatible features to participate in the mesh, in reality no provisions have been made for key IoT requirements such as a small footprint or low available system resources. Quite the opposite, 802.11s specification includes default mandatory routing protocol implementation for the HWMP routing protocol. In addition, the node can only be a member of either traditional Wi-Fi network BSS (Basic Service Set) or Mesh BSS, but never both. This has created a situation where 802.11s has been widely implemented only as a backhaul service for traditional Wi-Fi networks, rather than extending all the way to end user nodes. As such the 802.11s mesh networks are more relevant as device backhaul networks rather than as the access media for IoT device network connectivity. [13.]

2.4 Relevant (W)PAN technologies

2.4.1 ANT / ANT+

ANT and its ANT+ interoperability extension, created by Garmin, are a proprietary but open standard. They are intended as a standard to connect different health and sports related sensors and systems. ANT also includes a very flexible connectivity scheme enabling multiple different options based on the actual application needs as shown in Figure 1.

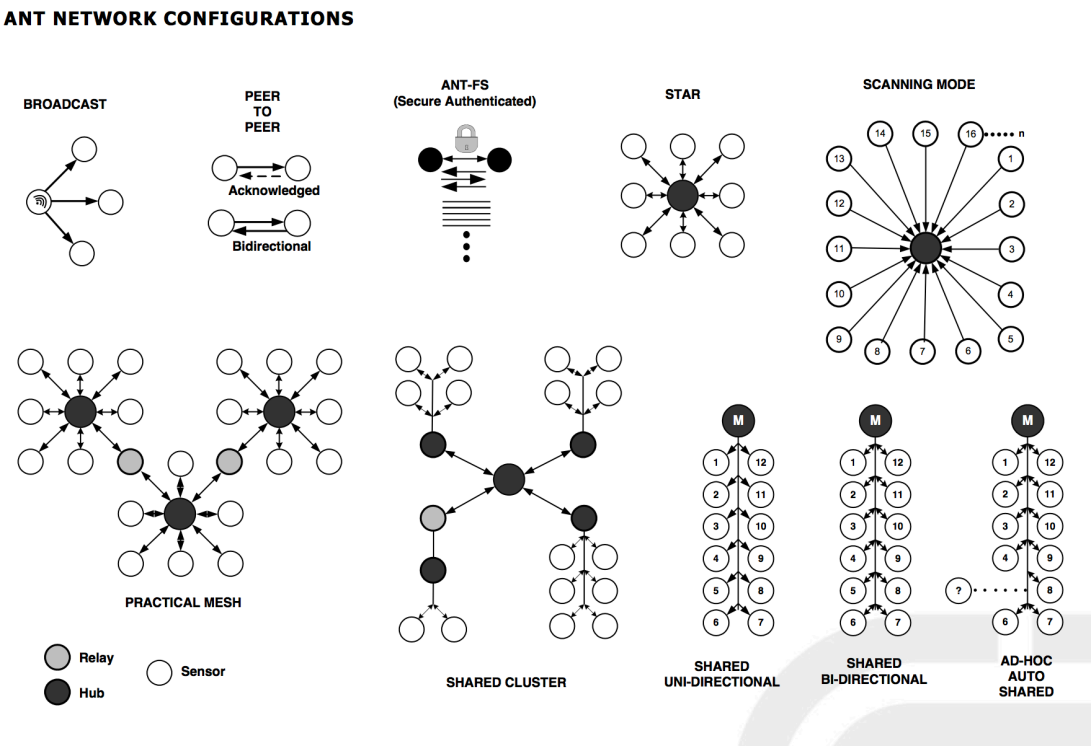

Figure 1: ANT network configurations. Copied from ANT Message Protocol and Usage [14].

While the ANT protocol is implementing connectivity for devices and things, such as bicycles and other sports equipment, which can be clearly seen as being in the core IoT field, it has clear focus on providing sensor data to measure the performance of individual users. In addition, to keep power requirements exceptionally low, allowing sensors to function years with off the shelf batteries, the ANT protocol has made several compromises in effort to keep the hardware simple. These compromises have resulted in allowing only very specific use cases for devices in the interest of providing

compatibility. This is made apparent by ANT+ device profiles that only allow for 19 different “device profiles” which each define a specific use case for a device using any of the profiles. While the underlying protocol can be used to implement applications outside the predefined profiles, there is no specifications-based interoperability between different devices. As such, ANT is not a general use IoT technology and will not be considered further in the context of this thesis. [15.]

2.4.2 Bluetooth

Bluetooth might be considered to be the original lightweight short distance wireless protocol to support low power sensors and other similar devices. In some cases, scatternet, as implemented by the original Bluetooth specification, is referred to as a mesh network. This is not entirely accurate and until Bluetooth LE implemented Mesh Profiles, no real mesh networking application existed in Bluetooth. [16.]

Bluetooth is one of the major IoT networking protocols today and it is further analysed in section 3.3.

2.4.3 Thread

Thread is one of the more widely supported mesh networking protocols today. It originated from Nest, which was later acquired by Google. However, today Google is only one part of Thread Group along with Apple, Arm, Nordic Semiconductor, STMicroelectronic and Texas Instruments.

As one of the major mesh networking technologies today, Thread is discussed in detail in section 3.4

2.4.4 Insteon

Insteon is a purely home automation focused communications protocol designed by a company called SmartLabs, which also holds all the rights and trademarks for the technology. One of the unique aspects of Insteon is that it utilises both wireline as well as wireless communication medium to create the network. This affords a certain amount

of flexibility to sensors and other nodes in connecting to the network, allowing for some unique use cases and increasing overall network reliability. Both wired and wireless links can form any part of the mesh and any node will also retransmit messages creating the mesh.

However, Insteon is inherently a closed system and the manufacturer has only chosen to make a limited API available for external interaction with the system. The protocol is also explicitly designed for simply relaying short control messages rather than any more complex data payloads. The Insteon network is only able to transmit at rates ranging from a few hundreds to maybe 1000 bps, which limits it usability [17.]. Additionally, SmartLabs have updated the system in 2014 with some more advanced features, but also fracturing the protocol at the same time, as they dropped compatibility to certain end user devices and older hardware [18.]. In-depth technical documentation is not available in the public domain and is only made available at manufacturers' discretion. Due to these factors, this protocol is not considered further in the context of this thesis.

2.4.5 ISA100 Wireless

ISA100 and its 101.11a sub-standard define a communications protocol for industrial applications. The intention of this standard is to create a highly robust, flexible and secure network to replace and/or augment existing wired instrumentation networks in process industries. The original standard was approved in 2012 for the first time and included mesh networking as one of the key methods to increase network reliability.

ISA100 standards also have a very tight focus on specific industrial applications due to their highly critical nature. Manufacturer devices are required to go through rigorous testing to ensure their compliance with the standard specifications. The protocol is not intended for use outside the industrial field, and as such the ISA100 standards governing body, ASCI (Automation Standards Compliance Institute) or the specifications are not suitable for use outside this specific field. This is made abundantly clear by the standards official goal setting as articulated by ASCI in their official materials:

> ISA100 Wireless (ISA100.11a / IEC 62734) is an international, industrial wireless networking and communications standard engineered to serve the needs of process industries. With native IPv6 networking and object architecture, ISA100 Wireless extends the Industrial Internet of Things (IIOT) to wireless. [19.]

Information on the protocol is not widely available, and also devices are available only to industrial end users. Additionally, the ISA101.11a cannot be considered to be of general use to IoT networking technology. Thus, it will not be further discussed in this thesis.

2.4.6 MiWi

MiWi is a proprietary WPAN protocol developed by Microchip with a focus on very low cost/low power applications. It is only available from Microchip directly as an integrated chip package with associated software, and none of the specifications are available in public domain. Mesh networking is advertised as one of the main features, but any further information is not made public.

MiWi has also been on the market for more than 10 years, but still the manufacturer does not advertise any major deployments or reference implementations. Quite the opposite, even the manufacturer seems to be more focused on alternative technologies such as Bluetooth LE, LoRa, 802.15.4 radio and Wi-fi, at the expense of MiWi [20.]. This protocol is included for the sake of giving a comprehensive description of the field in this thesis but will not be further discussed.

2.4.7 WirelessHART

WirelessHART is based on the industrial standard HART protocol and can be considered as an extension to the wireless medium. The original HART standard provides communication for field instrumentation in industrial applications. In this sense, WirelessHART is fairly similar to ISA100, but one of the key differences is that, whereas the ISA100 specifications allow for any layer 7 application and thus are more flexible to user requirements, WirelessHART only supports HART standard based applications. This ensures device compatibility and interoperability between manufacturers but also makes the technology less flexible.

While one of the core features of WirelessHART is mesh networking, the protocol has the same issues with no publicly available information and not being a generic mesh networking technology, as ISA100 [21.]. No further analysis of this protocol is included in this thesis.

2.4.8 ZigBee

ZigBee is one of the more popular IoT networking technologies and very widely deployed with more than half a billion devices sold [22.]. ZigBee is further discussed and analysed in section 3.5

2.4.9 Z-Wave

Z-Wave is a wireless networking protocol with heavy emphasis on mesh networking. It was originally developed by a Danish company called Zensys, but the rights to the technology have changed hands several times over the years. Currently, Z-Wave is owned by Silicon Labs, but certification and licensing is managed by Z-Wave Alliance. It has grown in popularity over the years with over 2,600 available products utilising it today. Z-Wave is analysed and is discussed in detail in section 3.6.

3 Current state of the major IoT Mesh networking technologies

3.1 General overview

The in-depth part of this thesis is limited to the following competing protocols: Bluetooth, Thread, ZigBee and Z-Wave. Products based on these protocols represent the vast majority of currently commercially available devices utilising IoT mesh networking technologies. At the same time, each protocol has a different historical technical background and area of focus. These differences are discussed in the following sections.

To enable highlighting differences between the protocols, the standard 7-layer ISO Open Systems Interconnect (OSI) model is used. However, it must be noted that each protocol has very specific implementation for different functions on an individual functional layer and not all of them are fully OSI model compliant. At times, the specifications omit some functionalities entirely, sometimes aggregating the functionalities of several different layers into one sub-protocol layer. For example, Thread and ZigBee specifications do not strictly speaking include OSI layer 1 or 2, since they utilize standard IEEE 802.15.4 LR-WPAN radio, whereas Z-Wave and Bluetooth both define their own radio layer functionality. Similarly, none of the protocols include full OSI model compliant session or presentation layer implementation, in the interest of keeping required software implementations simple and lightweight. Pertinent functions from these layers are included in the overlaying applications or omitted altogether as unnecessary.

3.2 General enabling technologies

3.2.1 Radio frequencies

As previously noted, all of the discussed protocols are based on wireless radio frequency technologies. Common to all of these protocols is their use of Industrial, Scientific and Medical (ISM) radio bands. The benefit is that the use of these bands does not require any kind of licensing from the end user's part. However, there are regional differences as to which frequencies are available, in some cases necessitating the implementation of region-specific hardware.

In general, the 2.4GHz ISM band is used by all of these protocols with the exception of Z-Wave, but this band has some shortcomings when it comes to frequency congestion and general interference due to its wide popularity. In general, higher frequencies also suffer from poor penetration in physically congested spaces, which can be an issue for IoT applications. To counter this, all the protocols include the use of lower 800-900 Mhz ISM bands with a notable exception of Bluetooth. Table 1 summarises the different frequencies used by the protocols.

Table 1. Frequency bands in use by different protocols

Protocol	Frequency	Notes
Bluetooth LE Mesh	2401.5 – 2480.5 MHz (Global)	2.4 Ghz ISM bands only Same as BT Basic Rate / Enhanced Data Rate
Thread and Zigbee	2400 – 2 483.5 MHz (Global) 868 – 868.6 MHz (Europe) 902 – 928 MHz (North America)	Both Thread and Zigbee use 802.15.4 PHY
Z-Wave	Numerous bands between: 865.2 – 926 MHz	Z-Wave uses individual country and region-specific bands.

3.2.2 IEEE 802.15.4 LR-WPAN protocol

Both Thread and Z-Wave utilise IEEE 802.15.4 Low Rate Wireless Personal Area Network (LR-WPAN) for the layer 1 and layer 2 functionality. The IEEE802.15.4 protocol provides a very energy efficient and low footprint radio technology for upper layer applications and protocols to use. 802.15.4 is fully open and well-defined standard and, as a consequence, there is a wide availability of hardware and software implementations, enabling easy integration with other higher-level protocols and eventual products.

IEEE released the initial 802.15.4 specification in 2003, and it has been updated, amended and revised since. The latest main version is 802.15.4-2018 and while specific amendments add new functionalities and expand the specification to different use cases,

the basic specification is still backwards compatible and ensures hardware interoperability.

The main focus of IEEE 802.15.4 is to specify the low complexity/cost/power radio frequency protocol for data communication devices, but with the design parameters, the available bandwidth remains low. The devices are only able to communicate at speeds from 20-30 kbps to 250 kbps depending on the used frequency band. One of the newest amendments, 802.15.4x, enables 2.4 Mbps data rate. However, no hardware implementation using this amendment exists yet.

3.2.3 External connectivity

Internet connectivity is of course one of the absolute core features of any IoT service or product. Counter-intuitively, most of the IoT products or protocols do not actually provide direct IP connectivity, but rather require a border node to mediate any access into the local node or network. Only partial exception to this is Thread, which uses 6LoWPAN that includes a partial IPv6 stack, but still requires a similar border node to provide access control and translation services for the connectivity.

This is due to the requirement of keeping the footprint of any implementation very small to enable the very low cost and minimal power usage of devices. In addition, the requirements for IoT node connectivity differ significantly from any fully Internet connected node. For example, the need to support higher level protocols, such as TCP, UDP or ICMP, does not exist in IoT applications, since a single node usually has very limited functionality, for example simply collecting data from a sensor/sensors or the basic control/actuation of devices. Thus, any aggregation of features and functionalities can be achieved by a higher-level application that has basic reachability into an individual node. As such full IP connectivity with the associated complexity is not required.

Each of the protocols described/discussed in this thesis takes a slightly differing approach to providing public network connectivity and border mediation functions. These differences are shortly discussed in each individual section.

3.3 Bluetooth

3.3.1 Overview of Bluetooth

The Bluetooth protocol was originally intended to be a way for mobile phones to connect accessories and to other mobile/fixed devices. The Bluetooth protocol was never developed with pure IoT applications in mind. One such clear limitation was the maximum number of nodes which was limited to eight for a long period of time and which was only changed with later revision of the specification.

Today Bluetooth has evolved to a much more capable protocol family, and at the same time mobile phones and smartphones have become one of the key devices for accessing and using everyday IoT networks and applications. One of the features that have enabled this change was the addition of Bluetooth Mesh protocol in 2017, which was based on the earlier Bluetooth LE standard.

3.3.2 Bluetooth pre-LE

The original versions of Bluetooth, until version 4.0, were developed mostly by mobile phone manufacturers with a clear aim of providing better interoperability, more robust connectivity and enhanced data rates for their application. These were, however, of limited applicability for IoT applications. The main downside of this approach was that energy efficiency or a small implementation footprint were not significant design considerations. Due to this, IoT devices utilising early versions of Bluetooth never really gained much success and were very limited in scope, mainly providing access to individual sensor/control devices rather than allowing a larger network of devices to be accessed.

3.3.3 Bluetooth LE

In 2010 Bluetooth SIG introduced updated version 4.0 specification for the protocols. Part of this update was an entirely new Bluetooth Low Energy (LE) protocol stack with a clear focus on enabling very low powered devices and rapid link establishment, both of which are core requirements for IoT mesh devices. The Bluetooth LE protocol stack later became one of the building blocks for Bluetooth LE Mesh or BLE Mesh, which launched in 2017. The previous version provided devices with simple 1-to-1 connections and even

a hub-and-spoke model by multiplying the single connections, but only Bluetooth LE enabled broadcasting data without pre-established connectivity. This feature formed the fundamental building block later enabling BLE Mesh.

3.3.4 BLE Mesh

Previous versions of Bluetooth are mainly concerned with enabling device-to-device connectivity in a robust and interoperable fashion. BLE Mesh is a networking technology created to enable a large number of these devices to form a mesh network and communicate with each other. While newer versions of Bluetooth have always remained backwards compatible, new additions to the specifications require devices to specifically support them, even though the basic building blocks are pieces of the specification from an earlier version.

BLE Mesh is easily the newest specification out of the four protocols discussed in this thesis. The basic protocol stack specification was only published in 2017, but as with any protocol, true maturity is only reached later. By January 2019, only version 1.01 has been published for the core stack. It must also be noted that there is a relative dearth of officially certified BLE Mesh components. At the end of 2017, only 13 individual components had been certified and that number has only increased to 223 today. In reality, the number of actual products is much smaller because individual components in the same product family are individually certified, in effect multiplying the number of certified products. However, the real power of BLE Mesh is not the number of components that are natively supported but its implementation of a specific backwards compatible component that enables any relatively modern Bluetooth device to connect into BLE Mesh network.

In the current form, the basic use case envisioned seems to be lighting control. The basic concepts and models have been designed with this in mind. Although there is nothing that would limit the use of BLE Mesh for other purposes, most of the available hardware and applications are also for building lighting management. While this is one of the most popular IoT mesh technology applications currently, all the other protocols described in this thesis have a broader scope of applications available today. It remains to be seen if BLE Mesh will prove to be popular outside this initially envisioned field.

3.3.5 Basic features of BLE Mesh

The BLE Mesh network is a fairly low bandwidth and a relatively high latency protocol for low powered devices. It is intended for transmission of small control messages and/or for a small amount of sensor data. While node-to-node links can theoretically have speeds up to 2 Mbps, the network structure and basic specification significantly limit bandwidth available to a single node [23.].

Bandwidth and latency are also very strongly affected by network size. An increased number of hops has a negative effect on both. Even in the best-case scenario, where there is only a single intermediary hop and the payload size matches the packet size, the realised bandwidth is in low single digits in kbps. Similarly, while effective latency in an optimal case might be approximately 20-30 ms, adding further intermediary nodes effectively multiplies the latency at every hop. When this is coupled with segmentation-and-reassembly (SAR) overhead when the payload size increases, it is possible to have latencies in the magnitude of several hundreds of milliseconds or even seconds, for fairly small payloads.

While this is not in itself a clearly disqualifying issue, it is something to be kept in mind as the limitations are significant from current day technology standpoint. These limitations are also compatible with many IoT applications, where bandwidth and latency requirements are overridden by a small footprint, application compatibility or low power considerations.

BLE Mesh also relies on managed flooding to deliver messages in the mesh. This effectively means that all nodes with relay functionality will rebroadcast messages to all other nodes within range. Receiving nodes will then again rebroadcast the message unless they have previously received it or unless the message time-to-live (TTL) has been exhausted. These mechanisms, along with the Friend/Low Power Node friendship relationship, form the “managed” part of the managed flooding method. While this is an inefficient way to utilise available bandwidth, reliability increases since messages can take multiple paths to reach their destination.

3.3.6 Network elements of BLE Mesh

The BLE Mesh specification defines four different functional node types. Any node can have one or more of these functionalities at any time and they can be disabled/enabled when needed. For example, a node with the Relay function can also act as a Proxy and Friend node. Only Low Power Node is usually functionally limited and is intended as a very low powered device with a low duty cycle, allowing minimal power consumption. Other nodes are envisioned to have permanent (AC) power sources not limiting their power usage [24.].

Different node type functionality is as follows:

- Relay Node – a basic node type which, in addition to a normal transmit/receive function, also forwards messages originally transmitted by other nodes in the network.
- Proxy Node – a node with proxy functionality implementing a standard Bluetooth LE stack in addition to the BLE Mesh stack. This enables Proxy Nodes to function as an intermediary device between the mesh and any Bluetooth LE compatible device, such as a mobile phone. Proxy nodes expose a GATT or Generic Attribute Profile interface to compatible external devices, which enables them to interact with the mesh network.
- Friend Node – a mesh node that functions in tandem with Low Power Nodes. Friend Nodes store messages addressed to Low Power Nodes when they are unreachable. The relationship between Friend and Low Power Nodes is called a friendship.
- Low Power Node – nodes that have very low power available for functionality. Low Power Nodes can be sleeping or turned off for long periods of time to conserve energy and will only wake up intermittently to perform their intended function. The Friendship relationship formed with Friend Nodes enables them to be addressed outside these periods.

3.3.7 Networking stack and structure of BLE Mesh

BLE Mesh stack is built on the existing Bluetooth LE specification. All nodes must include this basic level implementation to be able to communicate with each other. The BLE

Mesh networking stack has a well-defined structure, as shown by Figure 2, with distinct functionalities for each layer described below.

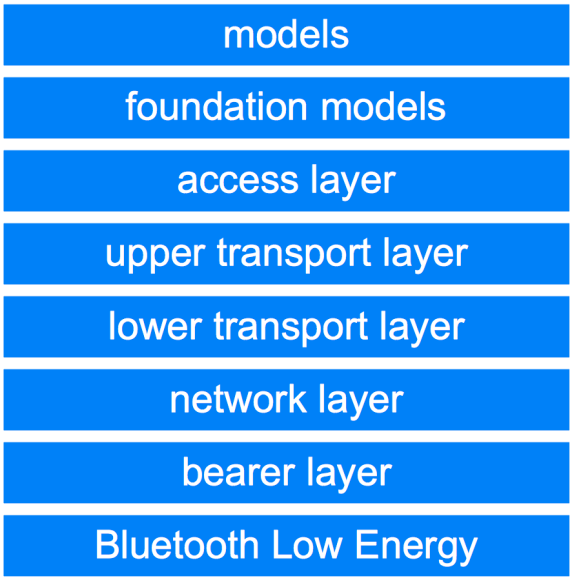

Figure 2: Bluetooth LE Mesh stack. Copied from Bluetooth Mesh Networking / An Introduction for Developers [24.]

Different layers include the following functions:

1) Bluetooth Low Energy:
   This layer is responsible for actual physical transmission of the data and abstracts some traditional OSI model layer 1 and 2 functionalities. Thus, any BLE Mesh implementation can remain physical media parameter agnostic, since this is all included and managed by the Bluetooth Low Energy part of the stack.
2) bearer layer:
   This layer is slightly non-consistent with the standard OSI model and simply creates additional encapsulation for higher level PDUs. It is used for backwards compatibility with standard Bluetooth LE devices that do not support the BLE Mesh protocol stack. This layer supplies two different bearers, Advertising Bearer and GATT Bearer. GATT Bearer encapsulates PDUs in a way that allows non-BLE Mesh devices to receive and transmit BLE Mesh PDUs and interact with BLE Mesh in limited capacity. One of the limitations is that they cannot be part of the mesh beyond communication with a single node.
3) network layer:
   The BLE Mesh network layer performs the more traditional functions of the standard OSI model. It defines the different message formats, network addressing and

performs basic filtering functions. It should be noted that BLE Mesh does not include any routing protocol functionality. This is also the layer where the Proxy and Relay Node functionality is implemented.

4) lower transport layer

   This layer partially implements functionality from the OSI model transport layer. The lower transport layer is only responsible for the segmentation and reassembly (SAR) of higher layer PDUs that do not fit into a single lower transport layer PDU.

5) upper transport layer

   The upper transport layer manages the secure access aspect of the BLE Mesh protocol. It is responsible for encrypting, decrypting and authenticating data passing through to and from the access layer. Additionally, some node-to-node control messages and management messages are generated on this layer. Friend Node functionality is also implemented on this layer.

6) access layer

   As the name suggests, this layer controls higher level application access into lower layers and implements OSI model presentation layer functionalities. The access layer defines how application data is formatted and verifies received data. The control of upper transport layer encryption/decryption functionality is also part of this layer.

7) foundation models

   While foundation models are described as one of the layers, they are not quite congruent with the standard OSI model. It defines special foundational functionalities for the network that are related to node provisioning, management and monitoring, but does not directly affect actual data transmission. The implementation of these foundation models is mandatory to any BLE Mesh device.

8) models

   The model's part of the BLE Mesh stack defines four predefined functions for the BLE Mesh network nodes. These categories are split into four general categories: generics, sensors, time and scene, and lighting. While these categories include several ready-made functionalities for the envisioned intended uses of BLE Mesh devices, their implementation is optional, and developers are free to define their own models.

3.3.8 External connectivity of BLE Mesh

BLE Mesh does not make any kind of provisions for native external addressability from general IP networks. Addressing, security schemes and lower level network limitations, such as PDU size, make direct external addressing entirely impractical. However, this naturally does not exclude the use of a separate device that is connected to both an external network and to the BLE Mesh. It can even be argued that the relative simplicity of BLE Mesh Model hierarchy makes implementing such a device relatively straightforward. Additionally, this makes BLE Mesh slightly more secure, since any translation of external messages would inherently include sanity checking and parameter verification.

However, BLE Mesh has a unique benefit and an advantage due to its backwards compatibility mode created with a special lower level bearer layer. Special GATT Bearer allows any Bluetooth Low Energy device to connect directly to BLE Mesh using Proxy Nodes implementing this functionality. This instantly allows almost any modern mobile phone, tablet or laptop computer to interact with the BLE Mesh network. Using a mobile phone as the platform for creating applications for end users to interface with the IoT device network has become the de facto standard and BLE Mesh has this support natively built into the protocol stack. A natural shortcoming of this approach is that the end user must be within reach of the network to be able to connect to it. While the short range of BLE Mesh might not be an issue for all applications, it is one of the limiting factors since allowing external connectivity would then require extra steps and devices to be included.

3.3.9 Energy consumption of BLE Mesh

One of the focus points for IoT applications is certainly low power operation of the network nodes. BLE Mesh implements a special Low Power Node (LPN) to allow devices with very low resources to participate in the network. The specifications allow the application to define the communication interval for the node which lowers the needed power even further. The interval may extend to hours or even days for applications that require only very infrequent communications between devices. In some cases, standard button battery can provide even multi-year lifetime for an individual LPN device. [25.]

Other node types in the BLE Mesh network are assumed to be permanently powered by an AC power source, and specifications do not include any provisions for lower power usage. Friend Nodes (FN), that enable LPN nodes to communicate only intermittently, must remain powered to ensure cached messages destined to LPN nodes, are not lost due to power loss.

3.4 Thread

3.4.1 Overview of Thread

The Thread mesh protocol is an open mesh networking protocol based on the IPv6 specification with emphasis on low power and security. The protocol is maintained and owned by Thread Group that makes the specification available free of charge to its members. Thread makes extensive use of other industry standard protocols: IEEE 802.15.4 for layer 1 and layer 2 connectivity and 6LoWPAN for IP connectivity.
Thread is a fairly new protocol having been published only in July 2015. It originates from Google's Nest product family but was significantly revised when made open and public. Current day Thread Group is a much larger ecosystem including, in addition to Google, some of the IoT industry's largest companies including ARM, Nordic Semiconductor, Apple, Amazon and Qualcomm. Thread Group also has a free academic tier making all the documentation available free of charge to students and educational institutions, which further encourages adoption.

The focus for Thread has been from the beginning to provide a lightweight IoT protocol using existing IP technologies. This makes connecting to existing networks painless and allows a low barrier of entry for developers that are familiar with associated protocol stacks. It also allows for products to be relatively future proof since the foundational technologies are very well defined and support is universal. In addition to this, Thread Group also has a certification program available to its members, which allows for testing product interoperability and protocol compliance, thus removing ambiguity from the eventual end user, ensuring that the product actually works.

3.4.2 Network elements of Thread

The main element of the Thread mesh is one Thread network. Networks are defined by their common security credentials and while a network can be divided into smaller entities called partitions, the security credentials are the same and retained when the network partitions. Each network has a single leader, and should the network partition, it will dynamically proceed to maintain this by electing a new leader node. When connectivity between partitions is restored, they will automatically merge into a single network again.

Individual building blocks for the Thread mesh are divided into two main categories, Full Thread Devices (FTD) and Minimal Thread Devices (MTD). FTDs are envisioned as permanently powered elements of the network, which can fully take part in forming the mesh rather than being simple client connected to it. MTDs are then the low powered or even temporarily powered end nodes that provide basic IoT functionality such as switches, actuators or sensors.

FTDs can have one of the following roles:

- Leader: Each network or partition has a single leader elected, which assigns router addresses and processes new router requests.
- Router: Forms the basis of the mesh network performing routing services.
- Border router: A special routing device that includes additional functionality to connect into other physical networks. The Thread network can have more than one border router to provide redundancy.
- Router Eligible End Device (REED): Router devices that can perform routing functions but are not currently acting as routers due to network topology or other conditions.
- Full End Device (FED): A normal Thread node that is only sending data through a parent Router or REED node.

MTDs can have the following functionalities:

- Minimal End Device (MED): A low power device that does not include any routing capability and communicates only through its parent node.

- Sleepy End Device (SED): A very low powered node with basic functionality. It will only communicate with its parent node intermittently. It includes polling functionality for queued communication.

The relationship between a router and end node is always parent-child relationship. The end node will attach to a single router or REED. The end node can have FED, MED or SED functionality but is always the child. The Thread node taxonomy is further clarified in Figure 3.

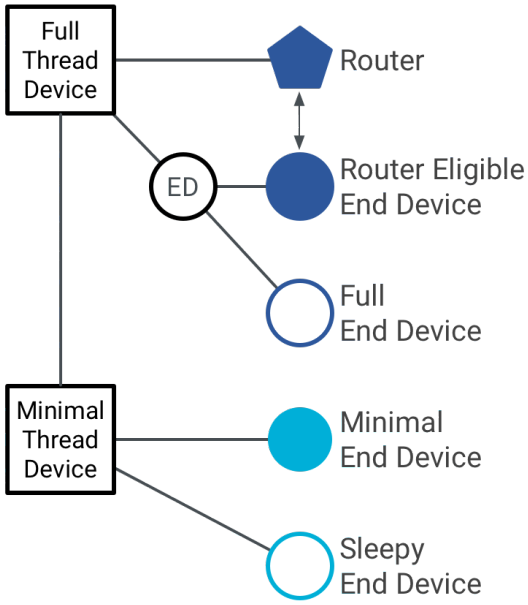

Figure 3: Thread node taxonomy. Copied from OpenThread – Thread Primer [26.]

3.4.3 Networking stack and structure of Thread

Unlike other protocols discussed in this thesis, Thread does not define application layer functionality and can actually accommodate multiple applications as long as the end node is able to process them. The Thread protocol provides a secure and transparent communications channel and a robust provisioning mechanism for network nodes, while staying application agnostic. Figure 4 shows the delineation between the underlying layer 1-2 protocol, the Thread core stack and the layer 7 application layer.

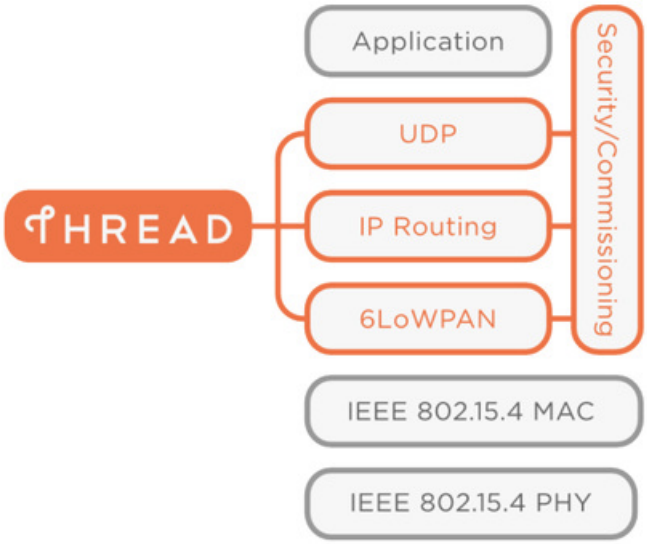

Figure 4: Thread networking stack. Copied from Thread - An Introduction [27.]

As previously noted, one of the core tenets of Thread is building the protocol on open and well-standardised building blocks. To this end, Thread uses the IEEE standard 802.15.4 for layer 1 and 2, allowing the usage of widely available standards-based radios and software components. Please see section 3.2.2. for further description of this protocol.

For layers 3 and 4, Thread more closely resembles normal IP networks. This is one of the original design goals for Thread since it enables direct connectivity to the Internet. However, Thread is purely implemented using IPv6 standards and since global IPv6 connectivity today is not available, one of the core functions for Thread Border Routers is providing translation service between IPv4 based networks and a local Thread mesh network.

For routing inside the Thread mesh, a simple RIP-like distance vector protocol is used. Thread borrows algorithms while omitting protocol specific message formats from the relevant RFCs. Thread node IDs also include basic topology information by including a parent router ID in the child ID. [28.]

Transport layer functionality is implemented using UDP. Since UDP does not provide for confirmed packet delivery via acknowledgements, similar functionality is implemented on

a lower MAC layer. Additionally, if an application has a strict requirement for packet delivery reliability, developers can implement any suitable mechanism on the application layer. The exception to this is Thread management and device provisioning traffic. Before the new node is allowed to join the mesh network or the network management application is used to access it, secure Datagram Transport Layer Security (DTLS) connection is first established to allow access.

Above the transport layer, Thread does not provide any services in the network and all functionalities beyond it are left for application developers to implement.

3.4.4 External connectivity of Thread

External connectivity is inherently one of the key functional aspects of any IoT product, and this service in the Thread mesh network is provided by specialised nodes called Border Routers. Border Gateways provide important service for the Thread mesh network, namely external connectivity, but they are not required for the network to function. The Thread mesh can be established and operated without any external connectivity and all aspects of it remain operational. In addition, one of the key points that Thread Group makes about their mesh protocol, is that it does not have single points of failure. Any gateway that connects to external networks is a such a point, but Thread supports multiple gateways for redundancy.

Border Routers also have some additional functions in addition to plain forwarding of the traffic. Since the Thread mesh network is inherently based on IPv6, Border Routers provide the necessary translation service for global IPv4 connectivity. In addition, basic firewall and rate limiting functions are usually implemented by Border Gateways to protect the relatively low bandwidth of the Thread mesh from malicious external or even site local traffic.
Beyond these network-focused services, Border Routers can also implement additional features to improve user experience with different applications. For example, different application-specific service discovery mechanisms can be created to allow more seamless integration of devices into a Thread mesh network. Another example is application-specific proxy service allowing access to sleepy SED node data through caching of the said data and without waking up the node, thus saving energy.

3.4.5 Energy consumption of Thread

The Thread protocol was originally designed for IoT applications and as such low energy consumption is one of the key features. Most Thread node types are envisioned to be permanently powered using an external power source, with the exception of SED nodes. While the MTD nodes can also be considered for low power application, the focus is more on a small footprint for hardware and software implementation and continuously powered operation is assumed.

SED nodes, on the other hand, can be very low power, to the extent that they can be powered by simple button batteries. For a simple sensor node connecting to the Thread mesh through its parent node, even this kind of minimal power source is able to provide sufficient power for several years [29.]. Achieving this kind of low power usage, of course, requires application developers to carefully consider their approach to communicating with individual nodes, but Thread and the underlying hardware platforms offer a good basis for achieving long device lifetimes.

3.5 ZigBee

3.5.1 Overview of Zigbee

The Zigbee protocol was originally developed alongside the IEEE 802.15.4 LR-WPAN protocol by Zigbee Alliance and was the first standard made available using this lower level protocol. The original Zigbee standard, which has since been obsoleted by newer versions, was published in 2005 as “Zigbee 2004 Specification”. Later on, a separate Zigbee PRO specification was published, which broke some aspects of interoperability with the standard Zigbee protocol family. In addition to this, Zigbee 2006 specification introduced the concept of Cluster Library, which is basically the application layer functionality built on top of the lower lever Zigbee protocol. However, these specifications and standards have changed several times and the current Zigbee 3.0 specification builds on all of these separate components.

In addition to the original Zigbee specification family, Zigbee Alliance has continued to build on the existing platform by publishing several other related technology standards that take advantage of and complement the Zigbee platform, such as:

- Dotdot – Replaces Zigbee Cluster Library for the IP network focus application layer, which can also be used with other competing IoT technologies such as Thread.
- rf4CE – Consumer Electronic layers 3-7 stack focused on a low device footprint.
- JupiterMesh – Industrial grade IoT mesh networking stack standard for layers 1-4 based on relevant industry standards.
- Green Power – Extension to Zigbee PRO standard allowing the use of ultra-low power devices with no external power sources, utilising power harvesting.

3.5.2 Network elements of Zigbee

Zigbee networks have three main functional elements: Zigbee Coordinator (ZC), Zigbee Router (ZR) and Zigbee End Device (ZED). These elements have very similar functions compared to other similar IoT mesh networking protocols. ZC nodes are responsible for coordinating the network and providing security services for node provisioning. ZR nodes provide message routing and parent functionality for child ZED nodes. The ZED nodes are then limited feature set low powered nodes that are able to communicate only intermittently with their parent node allowing for very low powered devices. ZC and ZR nodes are generally assumed to be permanently powered. However, one of the unique aspects of Zigbee mesh networks is the usage of beaconing networks. This option allows ZR nodes to also sleep and send a beacon to the network at preconfigured intervals. All ZED nodes will then wakeup intermittently to listen for this beacon to confirm network availability. While this would in general lower power requirements, it also requires accurate time keeping at ZED nodes, which might be counterproductive from a power management point of view.

3.5.3 Networking stack and structure of Zigbee

Zigbee uses IEEE 802.15.4 LP-WPAN for layer 1 and 2 services, Zigbee PRO for layer 3 and 4 services, and the Cluster Library provides the upper layers with an application focus. Zigbee takes a similar approach to BLE Mesh that also has a clearly defined application layer. The benefit of this approach is easier interoperability between devices from different manufacturers. However, this comes at the expense of implementation flexibility.

Zigbee has resolved this partly by allowing both private and company specific profiles in the Cluster Library.

In addition to this, Zigbee Alliance publishes several different protocol stacks under the Zigbee umbrella with different focuses dependent on the intended application. These build on the same underlying technologies such as the IEEE 802.15.4 protocol and partly on Zigbee PRO. These multiple protocol stacks are displayed in Figure 5, which clearly shows the flexibility of the Zigbee family protocols for different applications.

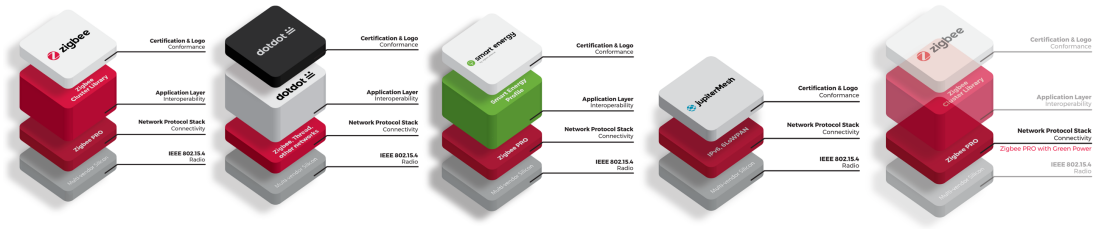

Figure 5: Different Zigbee networking stacks. From left to right: Zigbee 3.0, Zigbee Dotdot, Zigbee Smart Energy, Zigbee JupiterMesh and Zigbee PRO with Green Power. Copied from Zigbee Alliance website section [30.]

3.5.4 External connectivity of Zigbee

The Zigbee mesh network is not directly addressable from any other network and all connectivity relies on a special border router doing the necessary adaptation. Similar to BLE Mesh, this affords some security benefits since traffic filtering and sanity checking is inherently part of any translation process. However, the Zigbee 3.0 specification does not include any standardised way of implementing this kind of external gateway and developers are left with creating their own approach to connecting to any external services. There is certainly a large number of hardware platforms available, but today no simple way of connecting the Zigbee mesh to external networks exists.

3.5.5 Energy consumption of Zigbee

Zigbee 3.0 specification is designed from the ground up with low power applications in mind. While most network critical nodes including ZC and ZR are intended to be permanently powered, Zigbee does allow some flexibility in the form of beaconed networks. This allows even ZR nodes to sleep, lowering power consumption. ZED nodes, on the

other hand, are clearly intended for low power devices with limited resources. Nodes with very low duty cycles can remain dormant most of the time enabling multi-year battery life. New extensions allow Zigbee devices to function even without a primary power source only using power harvested from the environment or user action. [31.]

3.6 Z-Wave

3.6.1 Overview of Z-Wave

Z-Wave differs from other protocols in this thesis in that it was originally developed by a single company. Danish Zensys published the first version of the protocol that would later be standardised as Z-Wave in the early 2000's. It was based on their proprietary system-on-chip (SOC) product and at the time offered a unique combination of high performance and low cost. However, this approach was seen as too inherently dependant on a single company and in 2005 several industry players along with Zensys formed Z-Wave Alliance. The purpose of this organisation was to promote Z-Wave technology and ensure system interoperability between devices from different manufacturers. To this end, Z-Wave Alliance launched an official certification program in 2013.
One of the key differentiating factors for Z-Wave is that all the core intellectual property assets are owned by a single company and thus are not available to public. Silicon System, as the current owner of the original Z-Wave assets, has elected to make only small part of the core technologies publicly available. All implementations rely on SOC hardware from this single vendor. While this mitigates any lower layer compatibility issues, it leaves the whole ecosystem dependent on the technology choices that this single company makes. This closed nature of the ecosystem has been used as one of the arguments against larger adoption of Z-Wave.

Z-Wave, however, has had a significant early mover advantage having been an established technology for more than 15 years. It also has several unique technical attributes that are well suited for IoT deployments, such as exclusive use of lower frequency ISM bands in the 865-926 MHz range.

3.6.2 Network elements of Z-Wave

Z-Wave networks are relatively simple compared to the other protocols. It only has two basic node types called Controllers and Slave Nodes. The Z-Wave network also has a hard limit of 232 nodes in the network, due to original specification limiting Node ID to an 8-bit value. Today networks can be extended with a special bridge device if required, but this was only added as a feature in 2015.

A network can have several controller nodes, but only one of them can be Primary Controller (PC) at a time. The others will remain Secondary Controllers (SC) but one of them can take place of the PC node after network or node malfunction, in a process called healing. The PC node is responsible for administrating Network ID and allocating Node IDs during the node network joining process. The PC node also gathers, maintains and distributes the network topology information. Any Controller node must be permanently powered, but in addition to this, their position in the network must remain static, for them not to cause topology changes which then must be separately propagated.

Slave nodes are the Z-Wave standard nodes. They may repeat received frames, thus creating a mesh network, but they do not take part in network topology administration. Additionally, if a slave node repeats or routes frames not intended to it, permanent power is required. Z-Wave also has a low power feature called beaming, which allows low powered slave nodes to sleep for a set period of time and then wake up to process commands. These nodes are called Frequently Listening Routing Slaves (FLiRS).

3.6.3 Networking stack and structure of Z-Wave

The Z-Wave networking stack is superficially similar to the other protocols described in this thesis. However, there are also significant differences in individual layers. For layer 1-2, Z-Wave uses the ITU-T g.9959 standard short-range narrow band radio. The key difference is that the Z-Wave radio only utilises sub-GHz ISM bands, which affords it more range, but comes with a significant data rate disadvantage. Z-Wave raw data rates are fairly low at 100/40 kbps for the United States and 100/20 kbps for Europe and the rest of the world. Using region specific frequencies also requires separate hardware

implementations, but since there is only a single supplier for Z-Wave chips, this is less of an issue.

Layer 3-4 implementation is largely similar to other protocols, with a few notable features. Z-Wave layer 3 inherently includes deterministic message delivery with a mechanism to ensure retransmission of any partial or lost frames. In addition, Z-Wave is the only protocol described in this thesis to use source routing. This allows transmitting nodes to choose the best path through the mesh, but at the same time, it requires even low powered nodes to maintain at least minimal topology information.

One of Z-Wave's features enabling a high degree of interoperability is the usage of a standard application layer. The protocol includes a number of Command Classes specifying node functionalities and any node will declare which of these functions it supports when a network is being joined / is joined.

In general, the Z-Wave networking stack includes all the layers from 1 to 7 as displayed in Figure 6. This limits flexibility and reinforces the Z-Wave Alliance's message on their focus on providing interoperability above other technical attributes.

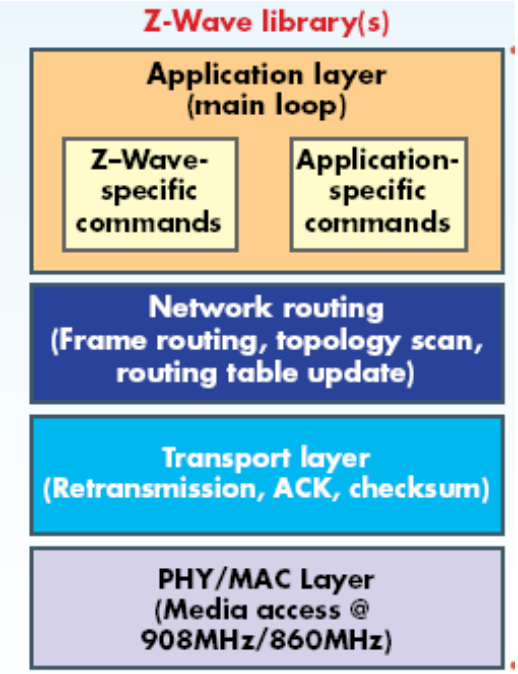

Figure 6: Z-Wave network stack. Copied from Application-oriented wireless sensor network communication protocols and hardware platforms: A survey [32.]

3.6.4 External connectivity of Z-Wave external

The Z-Wave mesh network is not directly addressable through any other common network infrastructure. One of the defining factors for Z-Wave devices is a very small device footprint and low power consumption which limits the functionality of individual nodes. Any device providing external connectivity to a Z-Wave network, must provide all translation and command interpretation functions to enable the control of any Z-Wave device. The protocol specifications do not include or contain provisions of any kind for facilitating external control.

3.6.5 Energy consumption of Z-Wave

Z-Wave devices are intended for low power applications, for example, in light switches and simple sensor devices. While most of the nodes are assumed permanently powered, low power nodes have support for sleeping and only transmit data periodically. This lowers their power consumption significantly. Newer generation Silicon Labs Z-Wave 700-series chips even have a decade long battery life with a normal coin battery. [33.]

3.7 Summary and comparison of primary attributes

3.7.1 Summary

All of the main protocols reviewed in this thesis are actively being developed today and have wide availability of systems and vendors. This does not mean that they are equal in all aspects. Their background and history, in addition to fundamental protocol design choices, make them different in several different ways. This section reviews and compares these differences from an overall systems point of view to give the reader some understanding of the strengths and weaknesses of each protocol.

3.7.2 Energy efficiency

Energy efficiency has been one of the fundamental enablers of a large number of IoT applications. The objects that are to be connected are largely not electronic devices nor even necessarily powered at all. All four protocols clearly take into account this kind of use cases. All include a node type that allows for extremely low power usage, mostly

through some intermittent transmission mechanism. While most protocols allow this mechanism to be controlled by the application, Thread uniquely leaves more control and also responsibility for the application developer since it does not include any application layer in the protocol. All the other protocols have set application layers or mandatory libraries of functionalities that set certain limitation for the usage. This kind of integration with the lower level systems can enable more efficient power usage.

All of the protocols enable basic IoT functionalities with very few power resources, for example with normal coin battery. In general, the lifetime of such a device can be measured in years or even more. The one improvement over this is Zigbee, which has an ultra-low power networking stack component that allows devices to operate even without an integrated power source at all, only using harvested energy.

3.7.3 The ecosystems

The number and wide availability of different devices and systems is of course important to end users.
BLE Mesh, due to its recent launch, has the smallest number of devices available today. This is partly offset by the ability of almost any mobile device to connect to the BLE Mesh devices. It is almost one of the most universally implemented networking protocols, making it a very attractive for developers.

Thread today has a fairly low number of devices available, but the fact that it utilises fundamental IP technologies makes it a very attractive platform. This is clearly shown by the fact that some of the largest consumer technology brands including Apple and Google are active members and contributors of Thread Group.

The same is also true for Zigbee, which has taken an open approach allowing almost anyone to access and implement their technology without major barriers of entry. Zigbee has also been well established in the IoT field and this is apparent from the number of devices and companies supporting Zigbee.

Z-Wave is the only protocol taking a slightly different approach. The system is strictly defined and there are significant cost barriers for entry. In addition to this, the Z-Wave Alliance has a single supplier hardware platform, not allowing any third parties to

implement fundamental Z-Wave technologies. Z-Wave Alliance has a long history in the field going back more than a decade. During the nascent years of IoT technology, the limited access approach was seen as a benefit due to guaranteed interoperability. This means that Z-Wave has significant existing support from the current install base and from the companies that have created product using it, but it is not seen as the most competitive platform anymore. Especially the announcement published in January 2020 that Amazon would be joining Zigbee Alliance was seen as a big loss for Z-Wave [34.].

3.7.4 Comparison of open and closed models

All of the protocols discussed in this thesis are managed and owned by central authorities with different models for ensuring technological viability. These entities generally own the core intellectual property and are involved in licensing relevant parts of it. This differs significantly from other Internet technologies, which are generally made available openly and without significant fees.

Z-Wave is closest to what could be termed as a closed technology model. Critical pieces of the technology are entirely closed and available only as commercial products, but also part of the software implementation is kept strictly accessible to consortium members for hefty fees. Small part of the application layer library was made openly available in 2017, but even that represented a very small part of the whole networking stack.

Beyond this, Bluetooth SIG makes almost all of the technology available to its consortium members, but fairly steep yearly fees are required. There are also specific licensing requirements and agreements that members must agree to. In general, Bluetooth SIG member is a corporation using BT in its products and due to historical reasons membership is heavily skewed to mobile phone and related product industries. However also due to Bluetooth´s wide availability, almost any platform has readily available BT libraries and software packages that make developing products easy.

Zigbee is slightly more readily available. Paid membership is required to access full protocol specification and other information, and also certifying products incurs separate fees. Zigbee has recognised this as one of their weaknesses and has made part of the stack available to public. In general, Zigbee licensing fees are less prohibitive and there

is a clear aim towards more open policies, while still ensuring core functionality and the brand not being diluted by poorly functioning products.

Thread is closest to what could be considered an open technology platform. When Thread Group was originally formed, one of the goals was to enable easier adoption of the specification, and to this end, Google published OpenThread. OpenThread is a freely available implementation of the Thread networking stack. In addition to this, the membership of Thread Group is available for free to certain developers and other fees are similarly very low. This makes developing applications for Thread easy, in comparison with all of the other protocols analysed in this thesis.

4 Example implementation using Thread

4.1 General description

The intention of this example implementation is to give an idea of complexity involved in developing a simple system to monitor environmental parameters in an automotive application. This is based on the author's need to obtain temperature data for several different storage spaces over a period of time from an example vehicle to facilitate safe storage of both automotive chemicals and consumer grade lithium-ion batteries, both of which are vulnerable to extreme temperature changes.

It should also be noted that in this test case, the specific vehicle allows the user to actively condition the interior space of the vehicle to allow persons and pets to remain inside the vehicle even while the vehicle is not being operated. However, the manufacturer's mobile application only provides a single temperature measurement for user monitoring. In this case the vehicle is what is commonly referred to as a "hatchback" model, in which the passenger compartment and trunk are one continuous space. The trunk is the area where pets are commonly kept, but the temperature of this space is not monitored at all.

In addition to this use case, the system could easily accommodate temperature monitoring for trailers connected to vehicles. In this case, having a probe connected to a mesh would be clearly beneficial, as other options, such as having wired sensors, would probably not be possible or would be extremely expensive.

In the test scenario, there are four separate interior spaces of which only two are effectively conditioned. In addition, the difference in estimated ambient temperatures during a single 24-hour test period exceeded 30 degrees Celcius, thus necessitating active measures to understand temperature parameters.

4.2 Test system

The tested system is based on the development boards of Particle.io, namely a single "Boron" LTE enabled mesh gateway and three "Xenon" mesh nodes. These boards use Thread to form the mesh network between nodes and are designed to easily enable

prototyping of IoT mesh networking products. All Particle.io products are tightly integrated into cloud services and all the communication to and from the created mesh network passes through their API layer. Integration through third party services is enabled by creating application specific event handlers which Particle.io calls “web hooks”.

However, one of the major downsides of the system in its current stage is that Particle.io does not make any diagnostic tools available for the mesh part. Mesh functionality was confirmed and simply deducted from device behaviour when individual nodes were repeatedly placed outside range and then an intermediary node was introduced into the topology by moving one of the other mesh nodes between the two disconnected nodes. Further tools and improvements are needed in the future, but they are not available at the time.

Beyond this, Particle.io provides good diagnostic tools to check the health of the non-mesh part of the devices. Several technical parameters are followed, such as signal strength, quality and connectivity latency. The user is also able to initiate checks in real time for any node if needed. The console also tracks history data, so that any trend and deviation can be gleaned from it. Figure 7 shows the device console view for the mesh LTE gateway node used in this example implementation.

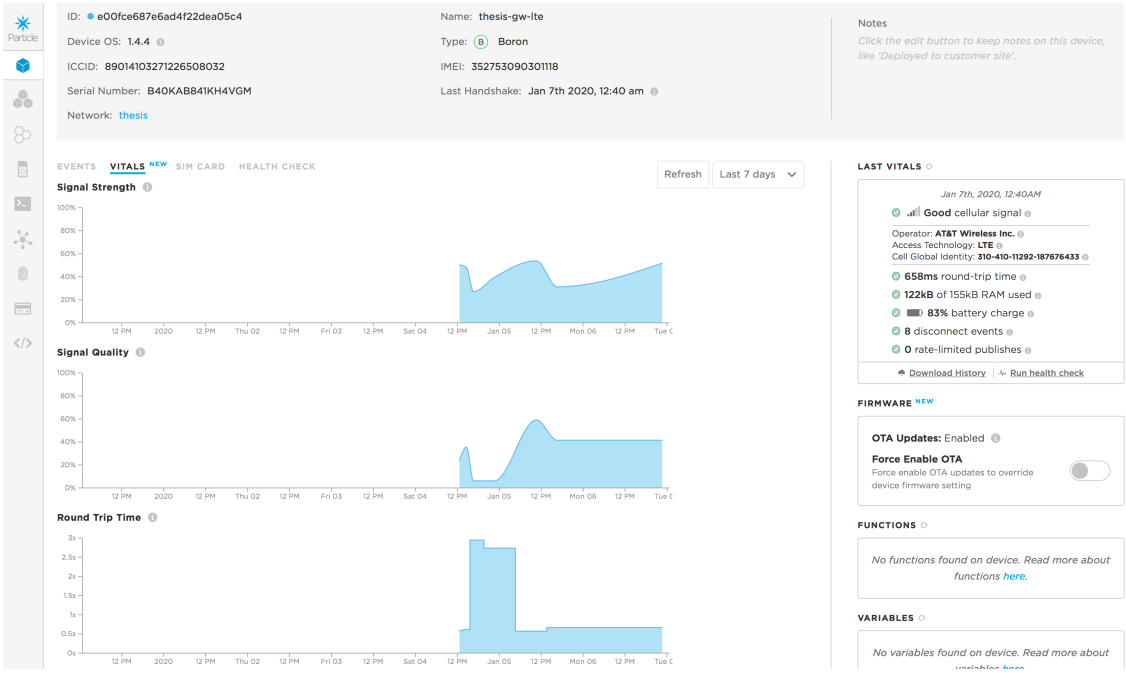

Figure 7: Individual console view of Particle.io. Screenshot [35.]

In the test system, the collected data points were pushed to the Ubidots cloud service, which allows for both easy integration with the chosen platform and also provides very flexible data analytics tools. Integration was achieved with the functions library of Ubidots which is compatible with the Particle.io devices.

The temperature measurements were done using a basic DHT11 type temperature sensor providing 1 °C resolution with +-2 °C accuracy. While not exceptional, these parameters were deemed acceptable for the demonstrated application. There is also a well-documented functions library available for the DH11 sensor on the Particle.io platform.

4.3 Network and device setup

One of the core features of Thread is an easy setup of mesh network nodes while maintaining good security. The Particle.io device setup process mirrors the suggested setup procedure of Thread where devices are recognised based on a physical QR code on the device and network access is authenticated through physical access to an existing mesh member device and the mesh password.

The setup and node configuration process is as follows:

Primary node and mesh setup:

Since a mesh cannot exist without at least a single device associated with it, it is usually created at the same time with initial device setup. The QR code is read with a mobile application to allow the user to access the device through Bluetooth. The mesh node is then placed in listening mode by pressing a physical button after which Bluetooth is used to setup initial configuration parameters for any other connectivity and to create the mesh network and associated password. During this step, the device is associated with the Particle.io user account which enables connection to their cloud services if external connectivity is available.

Creation of additional mesh nodes:

When a new mesh node is created, the initial process is the same. The QR code is read with the mobile application and the device is placed in listening mode. After this step, the user must place another node, which is already associated with the mesh network, in listening mode. After scanning the second device QR code, parameters for the mesh are automatically associated with the new device. The last step is for the user to provide the mesh password for the new device. This is to prevent unauthorised users that have physical access to devices from adding nodes to the mesh.

It should be also noted that Particle.io has a CLI tool available for more granular manual access to the same mesh setup process. Device QR codes are just used to simplify the setup process and allow the user to configure new devices without manually entering unique 24-character device IDs. This CLI based method only works when devices have external connectivity available and the devices have already been associated with a user account.

4.4 Particle.io integrated development environment

After the hardware setup, device programming and software management has been made simple through a web-based IDE environment. All devices sold by Particle.io are associated with cloud accounts for the service to enable the tie-in to a specific account and software. Non-configurable device firmware has a unique ID enabling identification of specific devices, which will then show up in the management console and development environment automatically.

This web-based development environment is also one of the key features of Particle.io devices in general. It is the main method for updating device programming over the air (OTA), and while the same process can be completed without connectivity, using a supplied USB cable and CLI software, it is clearly made the preferred method. When devices are not connected, software updates can also be queued so that an update is pushed to the device after connectivity is re-established.

In the IDE, the user is presented with a basic code editor, but the user is of course free to use any preferred editor. Particle.io uses C++ for programming, making it very familiar to anyone who has previously programmed comparable embedded systems.

IDE is very simple and only includes basic functions to compile, verify and push software into a single selected device. Also, a repository of different ready-made libraries is available free, in addition to any software and libraries previously created by the user. Links are available to separate management console and also to particle documentation as seen in Figure 8.

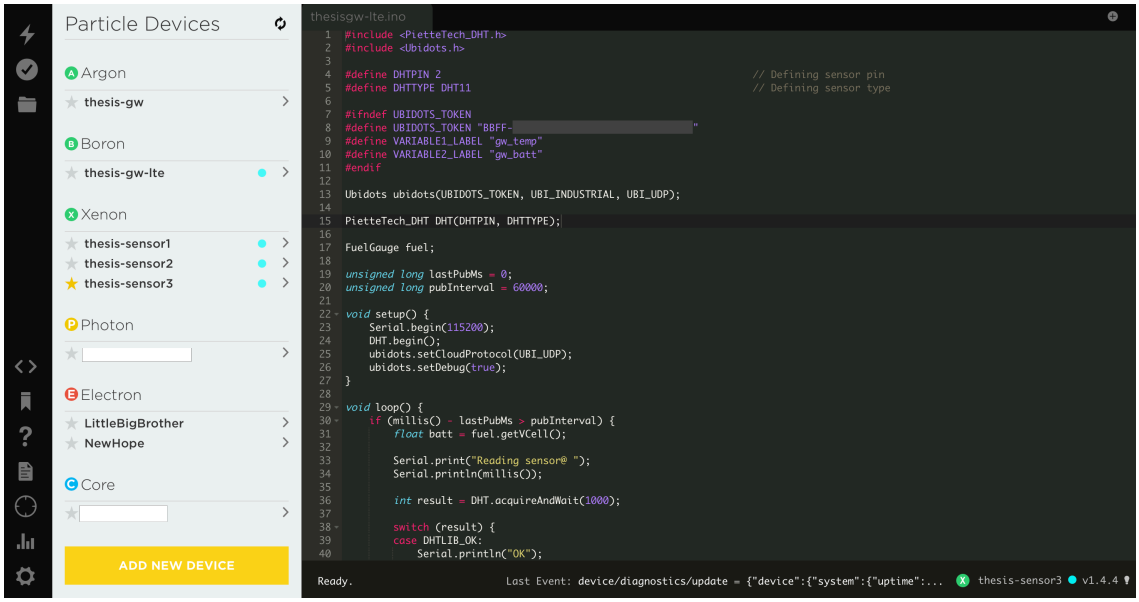

Figure 8: Particle.io integrated development environment showing the device selection menu. Screenshot [36].

4.5 Device programming and configuration

When the initial setup and device association with the used account was completed, all devices automatically became available in the IDE. A WiFi gateway device was used for development purposes, but it was replaced with the LTE gateway for the actual test data collection.

For both WiFi and LTE gateway devices, basic software was created where main loop simply listened out for any Thread Mesh publish events, which were then individually

pushed into the Ubidots cloud. The loop forked to read the local temperature sensor every 60 seconds and pushed that data into the cloud as well.

Software programming for the Mesh nodes was similarly very rudimentary, containing only main loop which published Mesh event containing collected temperature information every 60 seconds. In addition to the temperature information, also local battery supply voltage was read and pushed to Ubidots as well.

Both sensor integration and Ubidots connectivity were done using a ready-made library for the components. The biggest challenge was caused by low speed sensor chips, which required significant error handling to read reliably. This part was actually almost half of the actual code.

Overall, this approach to IDE seemed ideal especially as cloning software to individual sensors became a question of pushing the same software to each device effectively cloning the sensors. All device and event identification and sorting was entirely automatic and did not require any coding.

4.6 Ubidots cloud setup

Ubidots provides basic level free account access on trial basis to almost all functionalities with small “dot” counts. These dots are the individual data samples that can then be stored, analysed and further processed in the Ubidots cloud. Such an account was registered for the purposes of collecting the data for this thesis.

After registration, a single API key was created to access the Ubidots cloud from the devices. This API key was hard coded into all of the devices and it provides device authentication for the Ubidots cloud. When any event is pushed, it includes the unique device ID which then becomes automatically associated with all further data from the device. The Ubidots console displays all devices that have pushed data to the cloud using credentials associated with the account. Each variable pushed by an individual device is also listed in the console and these variables then become available for analysis, event triggering and visualisation using Ubidots tools.

In this case, a dashboard was constructed associating temperature variables from the gateway and all sensors into a single line chart and battery variables into individual battery charge level indicators. Ubidots is flexible in allowing customization of almost all aspects of these visualisation tools, including appearance, data selection and setting limits for individual variables. For example, the battery indicators have a four-level colour coding, allowing quickly identifying whether the device requires charging.

4.7 Test procedure and results

As indicated, the mesh is a combination of four mesh devices. One of them is an LTE modem which allows external connectivity. Each device was placed into a separate compartment in the vehicle to track the interior temperatures. One of the compartments is also a semi-effective faraday cage. Placing one of the sensors in the cage immediately allowed for an acceptable signal.

Figure 10 shows the overall sensor location in the vehicle. However, the devices were all placed in open, unclosed space in their respective compartments, to allow the sensor to accurately measure ambient temperature. The LTE gateway was placed in the main cabin to allow a good signal through the glass windows of the vehicle.

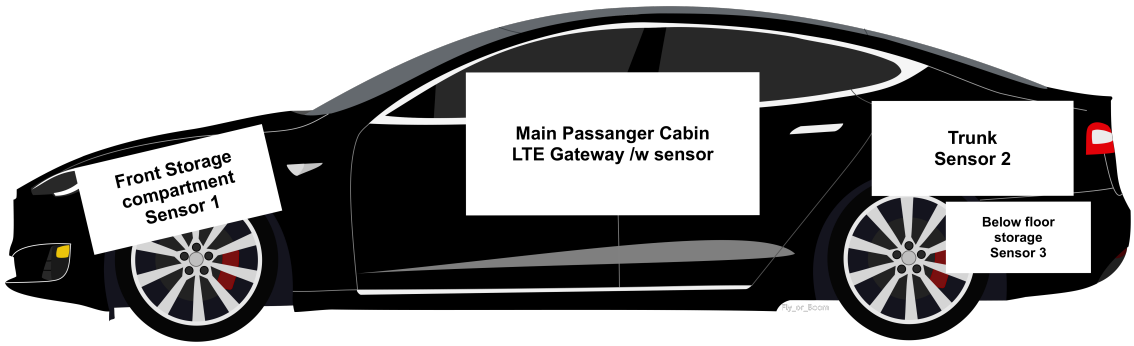

Figure 10: Compartment placement in the vehicle and the sensors located in each space.

Overall, the system proved to be very reliable and no major gaps were observed in the data samples, even in known poor signal areas. In addition, even with special power management code, all of the nodes had about half a week battery life with miniature 1000 mAh Li-po cells. The device duty cycle used was 1/60, and this seems to imply that multi-

month battery life would be achievable fairly easily with proper code optimisation and some additional logic in the sampling period, even with this limited power capacity.

The devices were left in the vehicle for 72 hours spanning normal use. The collected data had several interesting findings that could not be simply deducted from other information and had to be empirically tested. For example, the vehicle front motor quickly increased front storage space temperature after driving, due to residual heat transferring into the compartment. The lower rear storage did not have similar heating, even though the vehicle rear motor was similarly in close proximity. Figure 11 shows the collected data and the Ubidots dashboard that was created to track it.

Figure 11: Ubidots data dashboard used to track and analyse collected data. Screenshot [37].

5 Conclusions

IoT Mesh networked sensors are an ideal solution for collecting data in real time from challenging environments such as passenger vehicles. They provide a solution that would otherwise require a considerable amount of manual collection and processing of data from traditional data loggers or otherwise instrumenting a vehicle, which would be both time consuming and technically difficult.

The tested system, even in development form proved to be very functional and reliable in providing real time data. While the mesh network devices and protocols are quickly developing, even in their current form, implementing this kind of system was fairly straightforward and could be done with a minimal budget.

It can easily be imagined that this kind of a system could prove useful in monitoring a multitude of physical parameters or actuating a simple function. Especially when vehicles get larger than normal passenger cars, it becomes increasingly more costly to provide wired sensors for non-safety critical applications.

Beyond the maturing application field for IoT technologies, also the protocols themselves seem to be developing towards more uniform platforms. The recent announcement of the CHIP alliance has taken an approach similar to Thread's in that on base level the IP protocol was made transparent to applications. Since this alliance now includes many of the major IoT companies, including Google, Apple, Amazon and the Zigbee Alliance, it seems that there is a clear direction towards making IP truly a universal protocol. While the focus of this alliance is in connected home technologies, it can easily be seen as validation of the Internet centric approach taken beyond just this single application realm.

While all of the protocols in this thesis can be made to be compatible with the pure IP world, it is only Thread that previously had the IP protocol as one of the fundamental building blocks. Now Zigbee has also made it their choice. Inside the connected home application realm, there certainly seems to be need for a highly interoperable protocol that can be used in low power devices in conjunction with higher speed and higher power nodes. It remains to be seen whether Google's involvement in this latest development is problematic for Thread Group, since Google still has a strong influence on the project.

What is notable is that while BT has been one of the most well-known protocols in the IoT field for decades, it has not been able to transfer that footprint into the IP centric world. The same is also true for Z-Wave, which seems most at risk to be abandoned with further technological development.

Regardless of what the underlying technologies are, it seems indisputable that IP centricity is one of the core requirements for the future. It seems logical that the protocols that have been able to grow and include this in their fundamental nature are poised to be the platforms for tomorrow's applications.

References

1. Evans, Dave. 2011. The Internet of Things How the Next Evolution of the Internet Is Changing Everything. Cisco IBSG. Online. <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf>. Accessed 20 April 2020.

2. Sharma, Krishna Sharma; Bogale, Tadilo Endeshaw; Chatzinotas, Symeon; Wang, Xianbin, & Le, Long Bao. 2016. Physical Layer Aspects of Wireless IoT. 2016 International Symposium on Wireless Communication Systems (ISWCS). Poznan, Poland.

3. Brito, Jose Marcos Camara. 2016. Trends in Wireless Communications Towards 5G Networks — The Influence of e-Health and IoT Applications. 2016 International Multidisciplinary Conference on Computer and Energy Science. Split, Croatia.

4. Standards for the IoT. 2016. Online. 3GPP. <https://www.3gpp.org/news-events/1805-iot_r14>. Accessed 20 April 2020.

5. Sigfox Connected Objects: Radio Specifications. 2020. Online. Sigfox. <https://build.sigfox.com/sigfox-device-radio-specifications>. Accessed 20 April 2020.

6. Paret, Dominique; Crégo, Pierre. 2018. Wearables, Smart Textiles & Smart Apparel. London: ISTE Press - Elsevier

7. Sembroiz, David; Ricciardi, Sergio & Careglio, Davide. 2018. A Novel Cloud-Based IoT Architecture for Smart Building Automation. In: Ficco, Massimo & Palmiere, Francesco (ed.). Security and Resilience in Intelligent Data-Centric Systems and Communication Networks, s. 215-233. London: Academic Press.

8. Wael Ayoub, Abed Samhat, Fabienne Nouvel, Mohamad Mroue, Jean-Christophe Prévotet. 2018. Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs standards and Supported Mobility. 2018 25th International Conference on Telecommunications (ICT). St. Malo, France.

9. Weyn, Maarten; Ergeerts, Glenn; Wante, Luc; Vercauteren, Charles & Hellinckx, Peter. 2013. Survey of the DASH7 Alliance Protocol for 433 MHz Wireless Sensor Communication. International Journal of Distributed Sensor Networks.

10. Asovsky, Victor; Machani, Yaniv. 2016. Wi-Fi mesh Networks: Discover New Wireless Paths. Online. Texas Instruments. <http://www.ti.com/lit/wp/swry024/swry024.pdf>. Accessed 20 April 2020.

11. 4N 5G Americas White Paper: Cellular V2X Communications Towards 5G. 2018. Online. 5G Americas. <https://www.5gamericas.org/wp-content/uploads/2019/07/2018_5G_Americas_White_Paper_Cellular_V2X_Communications_Towards_5G__Final_for_Distribution.pdf>. Accessed 28 May 2020.

12. Connected Car Technology: Cellular V2X Outperforms DSRC/ITS-G5 in Comprehensive Tests as Mobility Industry Moves Towards 5G. Online. 2018. 5GAA - 5G Automotive Association e.V. <https://www.prnewswire.com/news-releases/connected-car-technology-cellular-v2x-outperforms-dsrc-its-g5-in-comprehensive-tests-as-mobility-industry-moves-towards-5g-854688653.html>. Accessed 20 April 2020.

13. Hiertz, Guido; Denteneer, Dee; Max, Sebastian; Taori, Rakesh; Cardona, Javier; Berlemann, Lars & Walke, Bernhard. 2010. IEEE 802.11s: The WLAN Mesh Standard. IEEE Wireless Communications 17(1): 104-111.

14. ANT Message Protocol and Usage. 2014. Online. Garmin Canada inc. <https://www.thisisant.com/resources/ant-message-protocol-and-usage/>. Accessed 28 February 2020.

15. ANT+ Device Profiles. 2020. Online. Garmin Canada Inc. <https://www.thisisant.com/developer/ant-plus/device-profiles>. Accessed 20 April 2020

16. Persson, K. E.; Manivannan, D. & Singhal, M. 2005. Bluetooth Scatternets: Criteria, Models and Classification. In: Ad Hoc Networks, Volume 3, Issue 6, s.777–794. Lexington, USA.

17. Irwin, David; Wu, Anthony, Barker, Sean; Mishra, Aditya; Shenoy, Prashent & Albrecht, Jeannie. 2011. Exploiting Home Automation Protocols for Load Monitoring in Smart Buildings. Online. University of Massachusetts Amherst. <http://www.cs.williams.edu/~jeannie/papers/insteon-buildsys11.pdf>. Accessed 28 February 2020.

18. Supported Insteon Devices with the Insteon for Hub App. 2015. Online. Insteon. <https://www.insteon.com/support-knowledgebase/2015/3/23/insteon-for-hub-supported-devices-by-platform>. Accessed 28 February 2020.

19. ISA100 Wireless Applications, Technology, and Systems. 2014. Online. Wireless Compliance Institute. <https://isa100wci.org/en-US/Documents/White-Papers/White-Paper-ISA100-Applications-Technology-and-Sys.aspx>. Accessed 28 February 2020.

20. Microchip Technology Inc. - Wireless Connectivity Solutions. Online. Microchip Technology Inc. <https://www.microchip.com/design-centers/wireless-connectivity>. Accessed 28 February 2020.

21. Nixon, Mark. 2012. Comparison of WirelessHART and ISA100.11a. Online. Emerson Process Management. <https://www.emerson.com/documents/automation/white-paper-a-comparison-of-wirelesshart-isa100-11a-en-42598.pdf>. Accessed 28 February 2020.

22. Analysts Confirm Half a Billion Zigbee Chipsets Sold. 2018. Online. ZigBee Alliance. <https://zigbeealliance.org/news_and_articles/analysts-confirm-half-a-billion-zigbee-chipsets-sold-igniting-iot-innovation-figures-to-reach-3-8-billion-by-2023>. Accessed 28 February 2020.

23. SI Labs – Mesh network performance comparison. 2019. Online. Silicon Laboratories inc. <https://www.silabs.com/documents/public/application-notes/an1142-mesh-network-performance-comparison.pdf>. Accessed 28 February 2020.

24. Woolley, Martin; Schmidt, Sarah. 2017.. Bluetooth Mesh Networking / An Introduction for Developers. Online. Bluetooth SIG.< https://www.bluetooth.com/wp-content/uploads/2019/03/Mesh-Technology-Overview.pdf>. Accessed 28 February 2020.

25. Darroudi, Sayed Mahdi; Caldera-Sànchez, Raül & Gomez, Carles. 2019. Bluetooth Mesh Energy Consumption: A Model. Sensors 2019: 19(5).

26. OpenThread – Thread Primer. 2019. Online. Google LLC. <https://openthread.io/guides/thread-primer/node-roles-and-types>. Accessed 28 February 2020.

27. Thread - An Introduction. 2014. Online. Thread Group. <https://www.threadgroup.org/Portals/0/documents/events/ThreadIntro.pdf>. Accessed 28 February 2020.

28. Thread Stack Fundamentals. 2015. Online. Thread Group. <https://www.silabs.com/documents/public/white-papers/Thread-Stack-Fundamentals.pdf>. Accessed 28 February 2020.

29. Azoidou, Eva. 2016. Battery Lifetime Modelling and Validation of Wireless Building Automation Devices in Thread. Online. KTH Royal Institute of Technology. <https://kth.diva-portal.org/smash/get/diva2:1067124/FULLTEXT01.pdf>. Accessed 28 February 2020.

30. Zigbee Solutions. 2019. Online. Zigbee Alliance website section. <https://zigbeealliance.org/solutions/>. Accessed 28 February 2020.

31. Zigbee Green Power Specification. 2014. Online. Zigbee Alliance. <https://zigbeealliance.org/wp-content/uploads/2019/11/docs-09-5499-26-batt-zigbee-green-power-specification.pdf>. Accessed 28 February 2020.

32. Pei, Zhongming; Deng, Zhidong; Yang, Bo; Cheng, Xiaoliang. 2008. Application-oriented Wireless Sensor Network Communication Protocols and Hardware Platforms: A Survey. 2008 IEEE International Conference on Industrial Technology.

33. New Smart Home Chip Lets Devices Have 10-year Battery Life. 2018. Online. The Verge. <https://www.theverge.com/circuitbreaker/2018/1/8/16839358/sigma-designs-z-wave-700-smart-home-chip-10-year-battery-life-ces-2018>. Accessed 28 February 2020.

34. Amazon is Taking More Control Over Smart Home Tech. 2019. Online. <https://www.theverge.com/2019/1/24/18196463/amazon-zigbee-alliance-board-smart-home-tech-control>. Accessed 28 February 2020.

35. Particle.io Device Management Console. 2020. Online. <https://console.particle.io/devices/>. Accessed 28 February 2020.

36. Particle.io Integrated Development Environment. 2020. Online. <https://build.particle.io/build/>. Accessed 28 February.

37. Ubidots Web Dashboard. 2020. Online. <https://industrial.ubidots.com/app/dashboards/>. Accessed 28 February 2020.