An Approach towards Decentralization of Systems Communication in Industry 4.0

To attain the academic degree of

Dr.-Ing.

Approved dissertation from
The Faculty of Mechanical Engineering,
TU Dortmund University

Aswin Karthik Ramachandran Venkatapathy

from

Coimbatore, India

Oral examination: 03 March 2021

Doctoral Committee:

1. Prof. Dr. Dr. h.c. Michael ten Hompel (TU Dortmund University)
2. Prof. Dr. Joseph Paradiso (Massachusetts Institute of Technology)

Dortmund, 2020

You, yes yourself
thank you!

Acknowledgements

I want to mention the two pillars of this work, who were also the supervisors: Prof. Dr. Dr. h. c. Michael ten Hompel and Prof. Dr. Joseph A Paradiso. They were instrumental in shaping the work to what it has become and steered me throughout the process.

Thanks to Mortiz Roidl, who identified my potential and believed in me, he made the possibility of becoming a scientist: a reality. Thanks to Jan Emmerich, who imbibed me with the technical experience and rigor to pursue new and bold endeavors with impervious logic behind it. Dr. Mojtaba Masoudinejad was a mentor throughout the scientific process and walked with me through every step.

Prof. Dr. Joseph .A. Paradiso took me in as one of his students and enabled a close collaboration with Ariel .C. Ekblaw and her research at Media Lab, MIT. I express my deepest gratitude as they believed in my skills as well as research and made the reality an unbelievable one by involving my work in experiments that were deployed in zero-gravity.

Through the research assistant position, financial support has been provided partially from the German Research Council (DFG) within the Collaborative Research Center (SFB876) *Providing Information by Resource-constrained Data Analysis* project A4.

Without Laura Hering, the work would have been impossible and incomplete without her unconditional good will and impeccable linguistic expertise.

Dortmund, November 4, 2020 *Aswin Karthik Ramachandran Venkatapathy*

Abstract

Decentralization of system communication of Industry 4.0 is the main thesis, where different types of system type topologies (Centralized, Distributed, and Decentralized) are explored. With the different topologies, the importance of systems decentralization in the context of logistics is explored for a socially networked industry. A socially networked industry is a hierarchical network of systems that collaborate and execute tasks with optimizations performed at every network level. To develop such a networking paradigm, decentralization is necessary, unfortunately it is not possible to provide the same strategies for all communicating systems. Therefore, the heterogeneous systems in logistics are classified broadly into two systems types based on the available communication data bandwidth and energy constraints. Upon classifying the systems into low power, low data-rate systems and high power, high data-rate systems, networking architecture for decentralized communication are developed. Two decentralized networking concepts are developed as part of this research work called the Decentralized Brains and the DezCom Context Broker, which is deployed in the industrial research facility of FLW, TU Dortmund. A sensor floor with 345 low power wireless sensor nodes is used to develop and evaluate the performance of Decentralized Brains, and a multi-robot system along with servers deployed in a cloud environment was used for analyzing DezCom context broker. Networking flooding primitive and decentralized networking software for low power wireless sensor networks were developed, and the codebase for Decentralized Brains is published. A guide for developing decentralized industrial applications is discussed with the advantages and disadvantages of developing decentralized applications for specific industrial use cases.

Keywords: Wireless communication, Wireless Sensor Networks, Systems Communication, Decentralized Systems, Context Broker, Industry 4.0

Contents

1 Introduction 1
- 1.1 Motivation 2
- 1.2 Research Questions 3
- 1.3 Structure of the Work 4
- 1.4 Problem Definition 6
- 1.5 Research Goals 7

2 Background: Decentralized Networking 9
- 2.1 Systems Communication and Nature 9
- 2.2 Types of Systems Architecture 12
- 2.3 System Type Definition 17
- 2.4 Why Decentralized Systems? 18
- 2.5 Approaches using Resource Discovery and Consensus 21

3 Decentralized Brains: Low-power, Low Data-rate Systems 27
- 3.1 Basics of WSN 27
- 3.2 PhyNetLab: Ultra-low-power WSN testbed 35
- 3.3 PhyNetLab Hardware Systems 44
- 3.4 The Sensor Floor 56
- 3.5 Decentralized Brains: Concept and Design 63
- 3.6 Conclusion 97

4 DezCom: High-power, High Data-rate Systems 101
- 4.1 What is a Context Broker? 101
- 4.2 Context Broker Architecture Standards from ETSI 105
- 4.3 Fiware - Orion Context Broker 108
- 4.4 MQTT 113
- 4.5 What is DezCom? 117
- 4.6 Inevitability of Blockchain 123
- 4.7 Problem Application Fit 125

4.8 Blockchain Technology 127
4.9 Decentralized Consensus Protocols 140
4.10 DezCom: A Decentralized Context Broker 146
4.11 Implementation of DezCom 150
4.12 Conclusion 152

5 Designing Decentralized Systems 155
5.1 Space Applications 155
5.2 Distributed Robotics 159
5.3 IoT Applications for Logistics 160
5.4 Design Guidelines 161

6 Conclusion 163
6.1 Summary 163
6.2 Outlook 164
6.3 Discussion and Future 165

List of Abbreviations 177

Appendix 181

Chapter 1
Introduction

Industrialization has evolved from mechanization to automation coming a long way from mechanical, agricultural machines to manufacturing domestic appliances on a large scale using automated production facilities. The paradigm shift of industrialization has been constant, and the next change is transferring research into the fourth industrial revolution, where the automated machines are equipped with intelligence for autonomy. The critical organizational benefit that leverages technological advancements is logistics. The operations and material handling are mechanized with autonomous machines with digital processes that improve the whole supply chain. One of the largest logistics service providers in Germany, DHL, attributes four key advancements for logistics and material handling [1] with the action of adapting to digital services. Transparency and integrity control along the whole supply chain is a factor that integrates a multitude of service providers into a single process with economic oversight and distributing business ownership across all stakeholders participating in the supply chain. As an industrial process spans across multiple disciplines of a supply chain, it becomes vital that a detailed real-time understanding of the status of the involved assets improves the efficiency of just in time production and optimizes resources across the whole industrial landscape. Integrity control of the goods allows for the trust-less integration of service providers. The digital processes allow for the checks and balances in quality control and accountability along the supply chain. This liberates the business owners while distributing responsibility across all stakeholders allowing for process optimization to shift from the center to the edge of the process where actual action and executive decision making are required. With all of the integrity, transparency, and accountability, a natural evolution towards fact-oriented transparent decision-making is enabled due to the vast amounts of data generated by digitizing processes and machines. These machines can

always leverage the available data to improvise execution in the field, whereas decision-making parameters are designed and modeled by the stakeholders. A central management system becomes obsolete as the supply chain's landscape transcends from the physical domain to the digital realm, allowing for hyper-scalable processes leveraging automated machines to function autonomously with the available data. Digital transformation liberates the devices and other participating entities in the supply chain network to orient away from a centralized architecture where decisions are made by a system acting as the central brain of the process. Such a liberation naturally evolves into decentralized systems where each system decides for the best of the network due to the transparency and integrity facilitated by the system. To allow for such transformation in the supply chain landscape, the involved decentralized entities and stakeholders should be able to communicate across the network transparently in a data-oriented approach. This decentralized orchestration of services allows for machines to have a dialog with the process to make local decisions optimizing the whole supply chain globally. This is the definition of *socially networked industry*, where intelligent computers, i.e., IoT devices, robots, and manufacturing machines along with business logic software, can freely communicate and negotiate on the execution of the process. The term socially networked industry also means the enabling of networked entities to discover voluntarily, associate, and interact in a network for achieving decentralized autonomy as targeted by Industry 4.0. Social networking is analogous to the term of a human social network, where humans discover other peers using a system to negotiate and work towards a goal. In machines, it is essential to allow for such a discourse. The devices can identify peers, discover their services, and negotiate to work towards a common goal in different parts of the supply chain.

1.1 Motivation

In this research work, one of the building blocks of autonomy is identified for which approaches and design specifications are created. Massive Machine Type Communications (mMTC) facilitating collaborative actions amongst physical machines and software processes to work in cohesion is the focus of this work. Precisely, the underlying communication and the ability to hyper-scale the network to span across the whole supply chain is the central theme. With scaling and high-level coordination among autonomous machines, decentralization becomes inevitable as there are heterogeneous machines that participate executing the process logic in the field. As different actions in a supply chain are abstracted into modular

processes for easy cohesion of multi-disciplinary systems to participate, process logic becomes the back-bone of any autonomous industry work flow. While decentralization is the focus for achieving autonomy in industrial processes, communication plays the vital role of connecting these machines seamlessly with as little human intervention as possible is the key enabler for the goals of Industry 4.0. Therefore, from the context of systems communication, reliable and robust communication systems are required that are decentralized by design to allow for existing products and processes in the supply chain to integrate into the network. The necessity of decentralization is also justified in 2.4, for designing and federating massively scalable, heterogeneous networks. As Industry 4.0 and machine autonomy is progressively adapted in the future, the critical feature of deployment will be the networking and communication between these machines. In this work, we try to enable machines to communicate in a decentralized network with focus on the communication and networking irrespective of the payload. Such a topic is chosen to enable resilient decentralized networking where machines can join and still run centralized applications without the bottleneck for a single point of failure. As decisions are abstracted from the centre and moved to the edges with critical decision making as a part of the hierarchical network, run-time optimization decisions are handed down to the machines involved in each process and the critical systems can be abstracted as reliable micro-services architecture. To allow for such decentralization and to enable a social discourse within the machines, we propose and design concepts for machine-based decentralized networks, which are then evaluated for performance in a real industrial scenario to be later applied in applications that might require these features. Here, the term *social* is lightly used in the context of machines having to organize and collaborate for executing processes in an Industry 4.0 context. All of the developed systems are available as design blueprints to be replicated for an actual scenario. When necessary, code repositories are also published to reproduce the results, which can be adapted according to the industrial applications.

1.2 Research Questions

The central question of this research is the decentralization of systems communication in Industry 4.0 with a particular focus on developing deployable systems that can be adapted to various applications depending on the industrial use case. Decentralization is the topical theme of this thesis, due to the trends in deploying massive IoT devices for process digitization and autonomous machines capable of edge intelligence. In this

thesis work, decentralization in communication is emphasized as it is considered the bottleneck of hyper-scalable IoT deployments. Part of the work is inspired by evolutionary biology and systems communication to abstract and transfer concepts from natural phenomena which is detailed in Sec. 2.1. It is not only used for transferring naturally occurring concepts but also drawing limitations that biological systems experience where computers perform better. Therefore, system development with the understanding of the paradigm shift in massively scalable decentralized machine organization in an industrial scenario is required.

Since there are low-power IoT devices and machines that are part of a heterogeneous network, this work focuses on a modular solution. It can be deployed depending on the application's requirements. A holistic, federated approach to developing the thesis is necessary to improve the adaptability of the developed systems. Therefore, evaluating the developed systems in industrial scenarios is another approach used for this application-based research along with developing reproducible and sharable systems that are not language-agnostic or platform dependent.

1.3 Structure of the Work

The work revolves around two pillars in developing future autonomous, self-organizing, highly scalable industrial systems. The work deals with deploying large-scale systems and focuses on decentralization of systems. These are two main goals, which are classified into two types of systems and provide two different kinds of solutions for the two broad systems classifications, which are presented in chapter 2. The description for the classification with the fundamentals of systems communication are detailed in chapter 2 and the whole work is divided into two parts as follows: (i) low-power, low data-rate systems (ii) high-power, high data-rate systems.

In low-power, low data-rate systems, the basics in wireless communication as well as the two testbeds that were developed as part of this work are discussed in chapter 3 in sections 3.1, 3.2, and 3.4 respectively. One of the testbeds was developed for experimenting on industrial wireless systems with a focus on energy-neutral systems. The second testbed is a low-power, low data-rate system with capabilities for low-power distributed sensing and rapid, iterative development of use cases. This testbed, called the Sensor Floor, is used further in the work for developing the concept of *Decentralized Brains* developed in chapter 3.

Basics of wireless communication and various testbeds for wireless sensor networks are described as the foundation for the design of these testbeds and then the problem of decentralized networking with consensus in low-

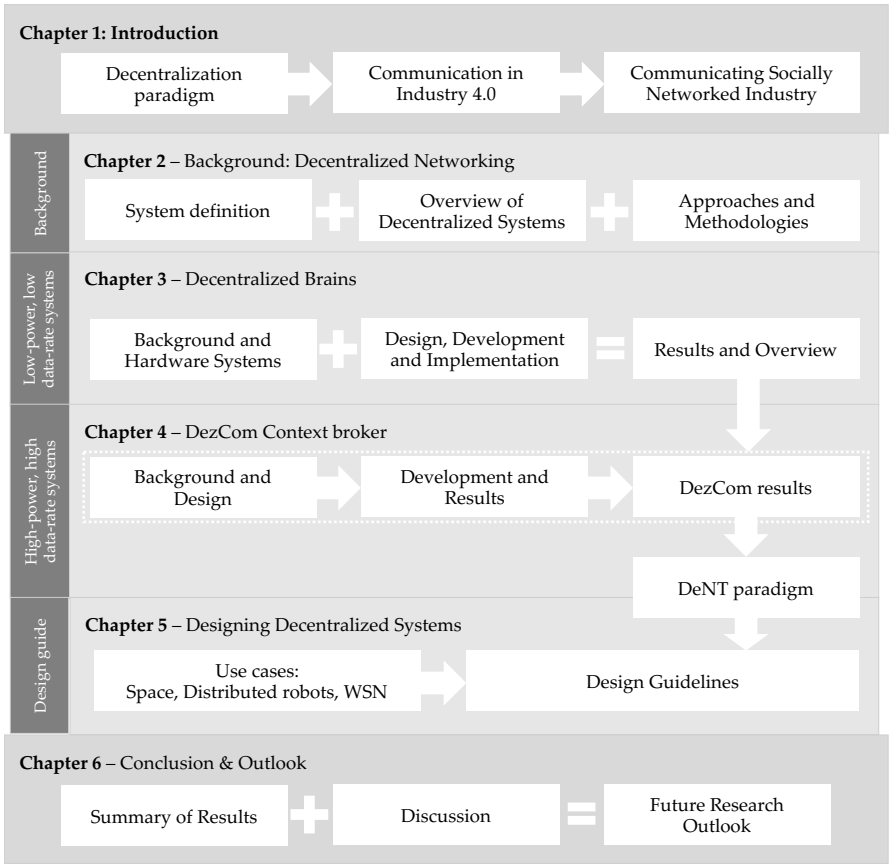

Fig. 1.1 Structure of work: Communicating the socially networked industry

power, low data-rate systems is conceptualized and a solution for such an application is developed in chapter 3. Testing, development and performance analysis are presented in the subsequent sections of chapter 3.

In part 2, high-power, high data-rate systems, basics of distributed communication are described and a specific solution for messaging called a context broker is discussed as a solution for communicating with high-power high data-rate systems in a hyper-scalable industrial network. The context broker is a communication system that is always available and relays messages between clients. In this case, the systems use a publish/subscribe model to send and receive messages. Two major context brokers are reviewed for the requirements in Sec. 4.1 and their features are considered as requirements for a decentralized content broker which is listed in Sec. 4.5.1. Using the derived requirements for a systems communication solution that is highly scalable, DezCom, a context broker

for decentralized communication is designed and developed. In this work, the basis for the solution is considered as blockchain technology for the context broker and the most imminent distributed ledger and blockchain technologies are studied in sections 4.8.1 to 4.9. A problem-application fit is found from the surveyed blockchain technologies in Sec. 4.7 and an implementation of the system is described in detail in Sec. 4.11.

In chapter 5, the basics for decentralized networking (DeNT) are described in Sec. 5.4 after exploring various use case solutions for diverse industrial applications. Finally in chapter 6, the outlook on developed decentralized systems and the primal contribution of this work are discussed with an outlook into future work. All of these chapters and sections are summarized as a block diagram in Fig. 1.1 that provides a holistic view on the work about *communicating the socially networked industry*.

1.4 Problem Definition

This application-based research work's central theme is to enable future systems to be developed as decentralized systems. The central hypothesis is that when systems are scaling and working autonomously, the backbone of such decentralized deployments are the networks and the underlying communication. Therefore, in this research work, two communication paradigms are presented in two parts: the necessity of hyper-scaling industrial networks and the requirement of autonomy that elicits communication to be self-organizing, decentralized, and collaborative. A proposal for the network architecture and reference implementation for decentralized networking are presented. The proposed decentralized networks are developed, tested, and evaluated in industrial scenarios. The evaluation is performed using a testbed for low-power systems and deployed in the cloud environment for high-power systems. The evaluation approach is chosen to develop a realistic assessment that can be carefully compared to actual industrial scenarios. The two parts are low-power, low data-rate systems and high-power, high data-rate systems. Since not all systems have the same system specification in-terms of available energy for operation or communication bandwidth, it becomes imperative to formulate decentralized networking solutions for specific scenarios. In this work, two systems are categorized as the networking paradigm that considers the two leading resource constraints, energy and communication bandwidth. As *one size fits all* type of solution is not possible in systems communication due to each system's resources, two cohesive networking paradigms are developed, which can be interfaced to function as a single hierarchical network. This work is a holistic approach to conceptualizing,

developing, and designing systems communication strategies published as reproducible results. The design includes modularity and an open-source development approach to proliferate the results' adaptation into the industrial application.

1.5 Research Goals

As there is an exponential rise in the number of battery-operated cyber-physical systems deployed in logistics, it becomes paramount to improve these low-power systems' communication paradigm. Additionally, these systems are always used along with high-power, high data-rate systems as access points, edge nodes, and fleet managers. The problem evolves into a two-part question that cannot be solved with a single solution. Especially in terms of decentralization of communication, as their communication and systems specifications are interlinked deeply. Therefore, in this research work, a two-part problem is proposed for which two different solutions for the decentralization of Industry 4.0 systems are developed and evaluated. The first part is where low-power, low data-rate systems are organized in a decentralized network using an efficient communication technique inspired by cephalopods' evolutionary biology called the Decentralized Brains. The critical issue of developing high density low-power wireless sensor networks is resource constraints. The limitations in such systems' resources are low-power as most of them are battery-operated, and the communication is performed using a shared medium. To overcome the aforementioned bottlenecks, a low-power medium access strategy is developed for low-power, low-data-rate systems.

The second part presents the organization of communication amongst high-power systems. In these high data-rate systems, energy or communication bandwidth is not the resource constraint. Instead, many members in the network and the underlying consensus to reliably function are the challenges. In an autonomous industrial network, there are multi-vendor systems that would collaborate in accomplishing tasks. Each system will depend on the information from a subset of scenarios that are deployed. As more information is generated, it is also necessary to understand the data's context to collaboratively self-organize and complete tasks. The nodes in the network, in order to perform autonomous organization and collaboration, should associate and disassociate with the system. They should be able to publish their actions, subscribe to available tasks, and access other involved systems. A decentralized network allows for the seamless integration of nodes into the network. However, such a network architecture introduces a host of challenges while deploying a

system that is considered as system requirements during system design. With the suite of requirements, relevant technologies are assessed to develop and evaluate the industrial facility system.

Chapter 2
Background: Decentralized Networking

In this chapter, communication in nature is explained and the evolution of communication across organisms over millennia is explored for design inspirations in Sec. 2.1. With the background of communication in nature and taking cues from evolutionary biology, the research work develops hyper-scalable solutions that are autonomously self-organized for collaborative industrial scenarios. For such an organization of systems within a network the various kinds of network topology, architecture and communication modalities are explored and solutions are proposed for specific problems. In Sec. 2.2, different types of systems with real world examples are provided which help in understanding the various system topologies and to provide the fundamentals for understanding the two part problem. The problem to solve along with the classification of the systems requires a two part solution that is designed and developed in this work. Therefore, the state of the art and background for each part relevant to the proposed solution and the systems used is provided in their specific chapters.

2.1 Systems Communication and Nature

In this section, we explore the nature of communication in evolutionary biology to understand the needs and aspects of communication. We do this to apply certain aspects of communication possible in evolutionary biology, not limited to humans. Since systems design is the outcome of the manifestation of imaginative human experiences, it is clear that all the communication aspects that were developed to this date were developed from the inspiration of humans experience. Communication primitives such as a server, client model, or communication topologies such as one-to-many,

peer-to-peer, have been designed from the experiences and possibilities of human communication.

Most intelligent biological organisms that can communicate context information and intent, such as the primates and vertebrates, communicate using sound. Marine mammals have the most elaborate modes of communication in the animal kingdom with sound as sound travels in water with ease and across a large area instead of in the air [2]. Due to these reasons, underwater acoustic signals evolved to be the principal mode of information transmission for fully aquatic mammals and a predominant mode of communication for amphibious marine mammals [2]. There are other modalities of communication, such as visual, chemical, tactile and acoustic signaling [2]. Acoustic signaling can be achieved by non-vocal or vocal signaling. A top-level predator has less non-vocal auditory communication, as in polar bears, as it may be adaptive to minimize non-vocal sounds [2]. Most of the organisms use acoustic, vocal communication using sound propagation as the primary medium of communication. An organ helps in modulating sound waves to produce distinctive sounds used to convey emotion and context. Communication in the animal kingdom has evolved to use sound. Since sound is a shared medium, only specific communication modalities are allowed, which is the most evolved and developed form of communication observed in nature.

Even though communication using sound has the same limitations as communication using electromagnetic radiation, specific parameters are overlooked. For example, communication in the animal kingdom, for instance, in humans, is limited by the processing speed of thought and the ability to control the organ for making very distinctive sounds. Moreover, the sensors in the organisms for sensing sound are limited in the sound spectrum, which is another parameter for perceiving sound communication between two organisms. The communication system is constrained from both ends, the selection of the medium, and the communicating entities. Therefore, there are also limitations in the communication that were developed with the perception of our experiences.

In this work, the main inspiration is evolutionary biology, but those inspired phenomena and experiences are not part of day to day human experience. In intra-species communication, using sound to communicate is the predominant mode of communication [2]. However, decentralization and communication among different actors are much more evolved within an organism than within a species. Decentralization has developed far better in organisms with common ancestors with humans from 300 million years ago [3]. This has escaped the perception of humans, and it is evident in the development of such systems [3]. These organisms have developed methods for integrating multiple intelligence centers into a single organism developing specific functional intelligence that decides on its own but

always in consensus with the whole entity. These are hints from evolutionary biology where bio-inspired decentralization of systems communication can be developed [3]. Even though they are not part of the intra-species communication, decentralization models can be used as inspiration for developing low-power, low-data rate decentralized wireless applications.

Moreover, in other sense, the biased thought in a centralized system design due to the limited perspective has limited many avenues where computers can function with ease and speed to manipulate physical phenomena. An example of a highly decentralized organism are the cephalopods that function with more than five brains, each specific to its avenue of intelligence, and can also understand and take action in consensus with other intelligence centers. The critical points on decentralization are inspired to develop a MAC strategy for wireless communication in Sec. 3, where efficient usage of a shared medium in decentralized communication is designed, developed, and validated. In Sec. 3.6, a proposal for developing a large scale decentralized industrial network is developed where the communication layer for the developed systems is abstracted as decentralized systems with the goal that every existing system is capable of retroactively adapting this kind of decentralized communication. To achieve decentralization on a massive scale, a consensus-based, distributed ledger is developed where every system in the network can agree on the published data concerning communication without considering the contents of the payload. Only the context of the communication is approved for further execution of tasks where each system can verify the validity of the contents.

In Sec. 3.6, the main goal of the context broker is to allow for even centralized systems to have communication in a decentralized network, which allows for developing new methods and adapting existing systems to become more manageable. In Sec. 3, the efficient usage of a shared medium is developed, and in Sec. 3.6, a large scale, the global industrial network is deployed where consensus is achieved in communication for developing new decentralized systems for industrial applications where adapting existing systems also becomes more natural.

At this point, omniscient consciousness is achieved across the participating systems through decentralization, which is still not a known phenomenon in biology. Computers communicating can choose to take part in the logic beyond the aspect of communication where every information is readily available in the network for processing but only on-demand with access levels federated by the industrial network consortium. When synthetic neurobiology can provide interfaces to the brain and a digital communication system [4], communication of knowledge through sound would be inefficient as consensus in communication can be achieved by

using systems as proposed in Sec. 3.6, at higher bandwidths using digital transmission.

2.2 Types of Systems Architecture

There are mainly two broad classifications of systems from the aspect of communication. They are centralized and decentralized systems with a practical evolution of a hybrid called distributed systems, which is also part of this classification. There is a thriving ecosystem in systems communication called the distributed systems in the path of evolving from centralized to decentralized systems.

In systems communication, we can postulate that distributed systems are a hybrid between centralized and decentralized systems due to the property that all decentralized systems are distributed systems but not all distributed systems are a decentralized system. Irrespective of the classifications, distributed computing and communication have gained economic relevance and wide-spread adaptation. Primarily, the skyrocketing application domain of cloud servers allowed the computing machines to be collocated across the globe and provided with the required fault-tolerance, robustness, and reliability in function and communication. This meant for internet applications to be deployed with high availability concerning geographical locations and on-demand scaling from serving a few hundred clients to a few hundred thousand in a few minutes. As intelligence moves from the center towards the edges, where the servers are not contained in high-security facilities, decentralization becomes essential.

Therefore, in this section, the two types of systems, along with distributed systems, are elaborated on to create the foundational elements for industrial systems communication, which is used in Secs. 3.5 and 4.11.

2.2.1 Centralized systems

We start with centralized systems because they are the most intuitive, easy to understand and to define. Centralized systems use client/server architecture where one or more client nodes are directly connected to a central server. Centralized systems are the most commonly used type of system in many organizations where the client sends a request to a company server and receives the response.

Example – Amazon. Consider a massive server to which we send our requests, and the server responds with the article that we requested.

Suppose we enter the search term 'batteries' in the Amazon search bar. This search term is sent as a request to the Amazon servers, responding with the articles based on relevance. In this situation, we are the client node; amazon servers are the central server. This is a typical example of a web service or an e-commerce platform that is part of a supply chain.

Characteristics of a Centralized System – Presence of a global clock: As the entire system consists of a central node (a server/ a master) and many client nodes (a computer/ a secondary), all client nodes sync up with the global clock (the clock of the central node). One single central unit: One single central unit which serves/coordinates all the other nodes in the system. Dependent failure of components: Central node failure causes the entire system to fail. This makes sense because no other entity is available to send/receive responses/requests when the server is down.

Scaling – Only vertical scaling on a central server is possible. Horizontal scaling will contradict the single central unit characteristic of this system of a separate primary entity.

Architecture of a Centralized System – Client-Server architecture. The central node that serves the other nodes in the system is the server node, and all the other nodes are the client nodes.

Limitations of a Centralized System – It cannot scale up vertically after a specific limit. After a limit, even if the server node's hardware and software capabilities are increased, the performance will not increase appreciably, leading to a cost/benefit ratio of < 1. Bottlenecks can appear when the traffic spikes – as the server can only have a finite number of open ports to listen to connections from client nodes. When high traffic occurs like a shopping sale, the server can substantially suffer a Denial-of-Service attack or Distributed Denial-of-Service attack.

Advantages of a Centralized System – Easy to physically secure. It is easy to secure and service the server and client nodes by their location. Smooth and elegant personal experience – A client has a dedicated system that he uses (personal computer). The company has a similar system that can be modified to suit custom needs. Dedicated resources (memory or CPU cores). More cost-efficient for small systems up to a specific limit – As the central systems take fewer funds to set up, they have an edge when small systems have to be built. Quick updates are possible – Only one machine to update. An easy detachment of a node from the system is possible.

Disadvantages of a Centralized System – Highly dependent on the network connectivity – The system can fail if the nodes lose connectivity as there is only one central node. No graceful degradation of the system – abrupt failure of the entire system Less possibility of data backup. If the server node fails and there is no backup, the data is lost right away. Difficult server maintenance – There is only one server node, and due to availability reasons, it is inefficient and unprofessional to take the server down for maintenance.

So, updates have to be done on-the-fly(hot updates), which is difficult, and the system could break. *Applications of a Centralized System* – Application development – Very easy to set up a central server and send client requests to the application.

Data analysis – Easy to do data analysis when all the data is in one place and available for analysis. Personal computing.

Use Cases – Centralized databases – all the data in one server for use.

Single-player games like Need For Speed, GTA Vice City – an entire game in one system (commonly, a Personal Computer).

Application development by deploying test servers leading to easy debugging, easy deployment, and easy simulation.

2.2.2 Distributed systems

These are the systems that have gained importance from the evolution of centralized systems to decentralized networks. It can be easily understood with the characteristic of transitivity i.e., all decentralized systems are distributed, but not all distributed systems are decentralized. Therefore, these kinds of distributed systems are presented as an intermediary step in networking, communication, and organization of entities in a process. Distributed systems gained importance in the data centers' realm, where a lot of redundant systems were deployed to assure the fault-tolerance nature of these systems. To keep all the systems in a network on the same page during execution and communication, developing methods would provide the needed reliance and robustness. Centralized systems were preferred to its straightforward nature of deployment, and the central issue of consensus was not necessary when the orders were centralized. The initial solutions were to provide consensus across the distributed systems by state-machine replication. The states of the network are propagated, agreed, and persisted across the nodes in the network. This was the first use for the development of distributed systems towards the goal of ubiquitous computing.

In decentralized systems, every node makes its own decision. The final behavior of the system is the aggregate of the decisions of the individual nodes. Note that there is no single entity that receives and responds to the request.

Example – Google search system. Each request is worked upon by hundreds of computers that crawl the web and return the relevant results. To the user, Google appears to be one system, but multiple computers work together to accomplish one task (respond to the results to the search query).

Characteristics of a Distributed System – : Concurrency of components: Nodes apply consensus protocols to agree on the same

values/transactions/commands/logs. Lack of a global clock: All nodes maintain their reference to time, called the system clock. Independent failure of components: In a distributed system, nodes fail independently without significantly affecting the entire system. If one node fails, the whole system continues to work without the failed node.

Scaling – Horizontal and vertical scaling is possible.

Architecture of Distributed System – peer-to-peer – all nodes are a peer of each other and work towards a common goal. Client-server – some nodes have become server nodes for the role of coordinator, arbiter. n-tier architecture – different parts of an application are distributed in different nodes of the systems, and these nodes work together to function as an application for the user/client.

Limitations of a Distributed System – It is challenging to design and debug algorithms for the system. These algorithms are complicated because of the absence of a common clock, so no temporal ordering of commands/logs can occur. Nodes can have different latencies, which have to be kept in mind while designing such algorithms. The complexity increases with an increase in the number of nodes. No standard clock causes difficulty in the temporal ordering of events/transactions. It is difficult for a node to get the global view of the system and make informed decisions based on the state of other nodes in the system.

Advantages of a Distributed System – Low latency than a centralized system deployment as distributed systems have high geographical spread, hence the communicating clients can choose a server that is closer to the client which reduces the latency.

Disadvantages of a Distributed System – It is challenging to achieve consensus. The conventional way of logging events by the absolute time they occur in is not possible here.

Applications of Distributed System – Cluster computing is a technique in which many computers are coupled to work to achieve global goals. The computer cluster acts as if it was a single computer. Grid computing – All the resources are pooled together for sharing in this kind of computing, turning the systems into a powerful supercomputer, mostly.

Use Cases – Distributed computing is a typical example of distributed systems in logistics, where real-time data is gathered, aggregated, and computations are performed in multiple computers. There are also examples of deploying a database service replicated across multiple servers for robust mission-critical operation in some cases.

2.2.3 Decentralized systems

These are another type of systems that have gained much popularity, primarily because of Bitcoin's massive hype. Now many organizations are trying to find the application of such systems. In decentralized systems, every node makes its own decision. The final behavior of the system is the aggregate of the decisions of the individual nodes. Note that there is no single entity that receives and responds to the request.

Example – Bitcoin. Let us take bitcoin, for example, because it's the most popular use case of decentralized systems. No single entity/organization owns the bitcoin network. The network is a sum of all the nodes who talk to each other for maintaining the amount of bitcoin every account holder has.

Characteristics of a Decentralized System – Lack of a global clock: Every node is independent of each other and has different clocks that they run and follow. Multiple central units (Computers/Nodes/Servers): More than one central unit can listen for other nodes' connections. Dependent failure of components: one central node failure causes a part of the system to fail, not the whole system.

Scaling – Vertical scaling is possible. Each node can add resources (hardware, software) to itself to increase the performance, leading to an increase in the entire system's performance.

Architecture of a Decentralized System – peer-to-peer architecture – all nodes are peers of each other. Not a single node has supremacy over other nodes. Master-secondary architecture – One node can become a master by voting and help coordinate a part of the system, but this does not mean the node has supremacy over the other node, which it is coordinating.

Limitations of a Decentralized System – May lead to coordination issues at the enterprise level – When every node is the owner of its behavior, it is challenging to achieve collective tasks. Not suitable for small systems – Not beneficial to build and operate small decentralized systems because of the low cost/benefit ratio. No way to regulate a node on the network – no superior node overseeing the behavior of subordinate nodes.

Advantages of a Decentralized System – The minimal problem of performance bottlenecks occurring – The entire load gets balanced on all the nodes, leading to minimal or to no bottleneck situations. High availability – Some nodes (computers, mobiles, servers) are always available/online for work, leading to high availability. More autonomy and control over resources – As each node controls its behavior, it has better independence leading to more control over resources.

Disadvantages of a Decentralized System – It is challenging to achieve big global tasks – No chain of command to command others to perform specific tasks. No regulatory oversight. It is challenging to know which node failed. Each node must be pinged for availability checking, and partitioning of work

has to be done to determine which node failed by checking the expected output with what the node generated. It is challenging to know which node responded – When a decentralized system serves a request, the request is actually served by one of the network nodes. Still, it is difficult to determine which node indeed served the request.

Applications of a Decentralized System – Private networks – peer nodes joined with each other to make a private network. Cryptocurrency – Nodes joined to become a part of a system in which digital currency is exchanged. The exchanges in the currency are without any trace and location of who sent what to whom. However, in bitcoin, we can see the public address and amount of bitcoin transferred, but those public addresses are mutable and hence difficult to trace.

Use Cases – Blockchain. Decentralized databases – Entire database split in parts and distributed to different nodes for storage and use. For example, records with names starting from 'A' to 'K' in one node, 'L' to 'N' in second node and 'O' to 'Z' in third node.

2.3 System Type Definition

This research work defines two categories for systems for decentralization of systems communication in Industry 4.0. Since one solution cannot fit all systems, the categorization is necessary to decentralize the communication for the self-organization of heterogeneous entities within an industrial network. The requirements of Industry 4.0 and logistics give us a guideline for such a categorization. The classification is based on the system specification of the entities that participate in the network. Two resource constraints govern the rules of classification. Energy is one criterion due to the explosion of IoT and the deployment of battery-operated devices in industrial processes. The amount of energy available also governs the amount of data that can be transferred for a process. Ultra-low-power wireless networked devices are battery-operated devices that process data from an event, sensor trigger or periodically. For example, a vibration sensor will trigger once a certain amount of vibration is detected. Until the trigger, the sensor monitors the physical phenomenon while the rest of the electronics standby in low-power sleep states. The amount of vibration is measured, and the CPU comes out of the low-power sleep state to decide if the acquired data is relevant for the process and if it needs to be communicated. These devices have ultra-low power energy requirements that operate over a long time and communicate minimal data. The second resource constraint is the communication bandwidth of the available data-rate. Even though energy is the main actor for this classification and

communication is dependent on the amount of energy available, the communication type plays a vital role while proposing solutions for systems communication. The first classification of the devices is a low-power, low data-rate device with a communication requirement that fits well within the IEEE 802.15.4 standard specification.

The second system type of classification is continuously powered devices with a source, it could still be battery operated, but the amount of energy available is considerably high with harvesting capabilities to sufficiently replenish charge. For example, an access point deployed outdoors with a photo-voltaic energy-harvesting allows the device to be always on/available and can communicate information from other low-power devices to the cloud. These devices are edge node or edge routers which facilitate communication of the low-power low data-rate devices. The resource constraint line is drawn at this point where the devices are highly available and have a large bandwidth to communicate information with context and actively participate in an industrial process.

2.4 Why Decentralized Systems?

This work hypothesizes that the decentralization of systems architecture will mitigate the problems and issues that arise due to machines' autonomous organizations across the supply chain. Machines are considered heterogeneous devices to software agents that can communicate, organize, and execute actions within a supply chain network. The following thought experiment gives us a theoretical explanation of why decentralization is the key to achieving autonomous organizations in a supply chain network giving rise to Decentralized Autonomous Organization (DAO). The following are the key reasons that answer the requirement for decentralization. Digitization has given rise to improved machine interaction where the machines are faster. The human nervous system is one of the most complex computational machines that can take numerous parameters across multiple disciplines to solve problems. However, once the question has been answered and needs to be repeated at an efficient pace, it is easier to transfer the computation into a machine algorithm executed by software agents. The speed of thought is a theoretical limitation where Prof. Tim Welsh of Cognitive and Neural Motor Behavior at the University of Toronto sets a limit considering multiple scenarios that it is 150 ms for a speed of thought calculated with a start as well as an endpoint from the stimulus to invoking the action. The example considered here is an athlete's actuation to start a 100 m sprint at the advent of a sound trigger generated by a pistol. Moreover, machines are faster than humans

and more efficient in their operational hours as they do not have downtime. The aspect of speed of thought is only considered without the accompanying complexities including the sheer volume of data that can be processed by the machines using GPU accelerated computational systems and TPU [5].

Furthermore, scaling these machines horizontally by replicating the same device and scheduling parts of the problem to be solved at different locations is another added advantage of using computational machines. The aggregation of the solutions can happen later once all the scheduled computations have been completed, as practiced in distributed computing. Efficient communication has allowed for such feats in the past with the evolution of cloud computing with geographically collocated data centers with specially hosted computational machines like Google Cloud's TPU products. When such intelligence shifts from the center towards the edges, where each system is self-sufficient with computational resources to organize itself in a network to execute tasks, the next question arises: *speed of execution*. When the speed of execution of a task surpasses humans as machines also have much lesser downtime than humans, we have seen a massive rise in the demand for autonomous machines. Here, with the help of intelligence shifting to the edges, self-organizing autonomous machines will identify, negotiate, and collaborate to execute tasks faster than humans by a minimum to an order of magnitude. We can safely claim such speeds in these communication, computation, and eventful organization scenarios when they happen within a localized network. This localized network works as part of the extensive supply chain across multiple locations, as the machines in the localized network work towards optimizing the system for its tasks, it is also optimizing the whole supply network. In this case of a localized network tasked for optimized operations, a centralized system would present a problem on two levels. *(i)* the transport of the sheer volume of data between collocated networks and *(ii)* a single point of failure that would halt not just the local supply chain but also the entire supply network.

As organizational units grow large, it is essential to mitigate the single point of failure. It is also essential to keep the communication among the systems flexible and robust. The network systems organize into units that collaborate by associating with the network and understanding the context in multiple locations. When systems require hyper-scaling of services, decentralization provides the necessary hierarchical network organization. Additionally, with autonomy and a socially networked industry, the machines communicate and negotiate to execute tasks that give rise to mMTC where decentralization can shift the intelligence to the edges to allow the nodes to function autonomously.

As large systems coordinate, multiple tasks are happening in parallel. It becomes paramount that a single point of failure or a networking bottleneck becomes the reason for the whole system not to function normally. Organizing the systems in a decentralized manner make them resilient against network failures and facilitate self-organization. These attributes also enable automated devices to complete tasks autonomously. These tasks are carried out without the central system knowing about the parallelized sub-tasks carried out within an Industry 4.0 scenario like the cyber-physical production systems. Some tasks are happening independently of each other, which requires communication and coordination between the participating systems, which is another reason for deploying these hyper-scalable systems as decentralized networks.

Summarizing the thought experiment for the hypothesis: *Why decentralized systems?*: speed of thought, execution, and the volume of data alone require the decentralization of the networks in a supply chain. Reducing human intervention by giving autonomy to the machines is the goal of Industry 4.0, where the speed of thought is a factor tackled by automation in execution of the task. Still, the intelligence of decision making is the critical enabler of autonomy which is hurdled by speed of thought. As decision-making is localized, it is essential to enable machines to communicate within the local network and collaborate on the decision-making process. Using decentralized communication enables the machines to request information that is not relevant to the system but important for the decision. Here, decentralized trust-less networks help authenticate and account for data and its access across the supply chain without a central federated service. All of these enabling features of decentralization rest on the shoulders of efficient, decentralized communication. Working towards the first steps of decentralization, it is identified that communication as the bottleneck considering the current systems only provides a centralized solution for communication both in the medium of propagation and data levels. A few of the aspects that are critical for achieving decentralization are discovery, consensus, and fault-tolerance. In discovery, a node identifies the network it has to join automatically. After joining a network, it can identify the node it has to talk to and discover its services to collaborate on tasks. Here, the essential criteria that keep every system synchronized are the consensus algorithms that allow every node in the network to accept that the information being communicated can be trusted and synchronized with the rest of the network. When there are heterogeneous machines in multiple locations that depend on specific information, consensus plays a vital role in synchronously evolving the network through time. Finally, fault-tolerance is crucial as it provides the necessary mitigation issues that arise due to discovery and consensus. Fault-tolerance strategies allow the network to function at an optimal level

even when there are service disruptions in the system. Therefore, this work focuses on communicating the socially networked industry across heterogeneous machines in the supply chain network. Consequently, the work also proposes approaches for decentralizing systems communication for Industry 4.0 applications.

2.5 Approaches using Resource Discovery and Consensus

The basic approaches for a decentralized network requires the nodes to communicate with protocols that allow for resource discovery, consensus models, and fault-tolerance. In this work, decentralization relies mainly on communication between systems. Therefore, the main goal is to create a fault-tolerant mechanism for a shared memory object or replicated state machines. In distributed computing, data replication is also known as log replication. The requirement is that all of the network nodes replicate specific amounts of memory queried by the system for decision-making processes. Systems and networks can achieve this through a consensus model that applies strict replication schemes with guarantees. The memory is highly available, always up to date, and consistent with all the other network nodes. Discovery is the initial mechanism with which any node interacts with the network. Following resource discovery, consensus models are another building block for decentralized systems that provide the necessary synchronous behaviors. The consensus models and resource discovery protocols may fail, and during these events of failure, it is necessary to mitigate or mask them for the nodes in the network. The strategies used in cases of failure to overcome the network's errors are called fault-tolerant behavior. Mitigation is a method in which the errors can be detected before their occurrence, and the errors can be prevented. In specific scenarios, mitigation is not feasible. In such cases, it is necessary to mask the errors for the nodes to function. Masking makes sure that the functioning of the network continues. Each of the necessary components of a decentralized network are described such as resource discovery, consensus models, and fault-tolerance in the following Sec. 2.5.1, 2.5.2, 2.5.3 respectively.

2.5.1 Resource discovery

Resource discovery is an essential feature for a decentralized network and for devices that are targeted to achieve autonomy. It is necessary that a

device can detect and join a network without or very little human intervention. For example, a Bluetooth device needs to be in the location, and both the devices are triggered to associate into a network. Once they are associated, they can discover each other using beacons and communicate. Resource discovery and description mechanisms for decentralized devices, service descriptions, and discovery in networking environments are necessary [6]. Resources can be anything in a decentralized network such as a device identifier, service identifier, network identifier, or location identifier. Resource discovery generates advertisements in a uniform description format that is broadcast across the whole network or a local sub-network. The consistent description format is a data payload understood by all other nodes to describe resources on a network. Device descriptions from additional resource description protocols and discovery methods can be used to generate the broadcast advertisements. The advertisements with this consistent resource identifier are used for cross-network discovery and access to the resources. An advertisement may include but is not limited to physical endpoints, virtual endpoints, user-extensible metadata, location information, and an UUID for the associated resource [6]. If a resource cannot provide an UUID, the mechanism generates an UUID for the resource. One manifestation of resource discovery in decentralized networks can have proxy nodes for generating advertisements for resources that cannot participate directly in the resource discovery and description mechanism. These proxy nodes may also serve as a proxy between local sub-network discovery and cross-network discovery of the advertisements [6].

Here comes the next vital factor, which is *network association*, which also plays an important role in machine autonomy. The device can either discover an existing authenticated network or form a network locally with a cluster of nodes that require communication.

2.5.2 Consensus models

Consensus models are another critical factor for decentralized networking as the devices do not have a central node to communicate. A centralized architecture provides the guarantee and stability required for a network. The nodes fail to communicate within the network if the single-point failure node is compromised, not only the specific node, but the whole system comes to a standstill. Consensus is a fundamental problem in fault-tolerant distributed systems [7]. Consensus involves multiple servers agreeing on values. Once they decide on a value, that decision is final [7]. Typical consensus algorithms progress when most of their servers are available; for

example, a cluster of 5 servers can continue to operate even if two servers fail [7]. If more servers fail, they stop making progress (but will never return an incorrect result) [7]. There are multiple proposals for consensus models in distributed systems that can be applied along with fault-tolerant behaviors to mitigate for errors that might occur. The Raft consensus algorithm is designed to be easy to understand. Raft's equivalent to Paxos in fault-tolerance and performance [7]. The difference is that it's decomposed into relatively independent sub-problems, and it cleanly addresses all major pieces needed for practical systems [7]. In this research work, we drive the research towards decentralized consensus models and the important state of the art consensus models and protocols are surveyed and studied in Sec. 4.8.2 and 4.9

2.5.3 Fault-tolerance

Fault-tolerance implements strategies for failures in the network. Fault-tolerance build a component as such that it can mask the presence of faults in the network. It helps mitigate the shortcomings imposed on a decentralized network due to the consensus model's failures or a leader election procedure. *Leader election* is required for all the nodes to follow a leader. Leader-less networks are purely decentralized networks where every node acts as a leader and a node in the system. When any node fails, the failure of the node does not affect the functions of the network. In distributed computing, the leader node provides authority for specific functions such as consensus or unique identifier generation. When a leader node is compromised, a reelection phase is triggered where all nodes can participate to become the leader. This allows for a distributed network to perform like a decentralized network. This provides stability and the necessary guarantees for a well functioning decentralized network as the network elapses over time and when few nodes disassociate and re-associate with the network. The leader provides the current status of the network. In the case of state machine replication, the leader node can authoritatively provide the latest information, after which the nodes can participate in the consensus rounds. These are the few essential features that are required for developing a decentralized networking paradigm.

There are also other features such as security in a trust-less architecture, scalability of the nodes as well as communication, heterogeneity of the nodes that are considered crucial while developing a decentralized communication framework or platform.

Decentralized Brains:
Low-power, low data-rate systems

The classification of low-power, low data-rate systems are battery powered devices for the majority of its operation. A biologically inspired concept is developed and evaluated to achieve decentralized communication in a low-power, low data-rate network called the *Decentralized Brains*. The *Decentralized Brains* design inspiration is as follows for developing a low-power decentralized industrial IoT system. From the perspective of evolution, the main reason for certain organisms, such as octopuses, to have separate brains or *Decentralized Brains* is to delegate motor processes without detracting from other vital functions. We find Neurons in the arms of an octopus, which can independently taste, touch, and control basic motions without the supervision of a central brain [3]. This decentralized intelligence allows for autonomous task completion—an intent is broadcast across the *brains*, and respective nervous systems take action locally. There are multiple instances in industrial contexts where similar notions of decentralized multi-agent systems in CPS, robots, and drones have been explored [8–11]. Collective behavior from a biological perspective often involves large numbers of autonomous systems interacting to produce complex assemblies [8]. Kilobots demonstrate the ability of self-assembly in a large-scale independent miniaturized robotic system by creating and programming swarm behavior to achieve a global behavior [8].

Swarm behavior is very much relevant in autonomous industrial robots where two automated machines communicate and collaborate to execute tasks. The application of low-power decentralized communication also extends to space applications, where robotics play a vital role in low-cost, independent human exploration. The growing interest in and proliferation of multi-part constellations in orbit, self-assembly of space assets, and swarm-based architectures shows that the space industry needs robust, low-power, and decentralized communication architectures to accompany distributed sensing capability. Space assets need to communicate with nearby neighbors in real-time, exercising communication protocols for a low data-rate while maintaining shared state awareness among all interacting hardware units.

Similar problems have been solved in the warehousing and shipping industry. Extensive logistics management necessitates keeping track of many items across physical space, across time, and across the robotic platforms that contact products [9]. Data replication in decentralized, low data-rate networks can facilitate a smooth operation and monitoring of collaborative maneuvers in space. In this part, low-power, low data-rate networking for decentralization is conceptualized and developed. The developed system is evaluated in a testbed called the Sensor Floor with 345 nodes. The results are presented and discussed with the practical advantages and disadvantages of constructive interference.

Chapter 3
Decentralized Brains: Low-power, Low Data-rate Systems

> Paradigm shift *is a fundamental change in the basic concepts and experimental practices of a scientific discipline.*
> *-Thomas Kuhn*

In this chapter, low-power communication using radio frequencies is the focus where a paradigm shift in the understanding is developed with the concept of *Decentralized Brains*. The concept of *Decentralized Brains* is detailed in Sec. 3.5. To understand the *Decentralized Brains* concept, two testbeds developed as part of this research work are detailed along with the fundamental concepts for wireless sensor networks and wireless communication standards for low-power, low data-rate systems. The two testbeds developed were the PhyNetLab and the Sensor Floor. The PhyNetLab is an ultra-low-power wireless sensor network testbed. Many testbed results and limitations gave rise to developing another dual-band, large-scale, high-density deployment called the Sensor Floor. The developed concept in this chapter, called the Decentralized Brains, is evaluated in the testbed.

3.1 Basics of WSN

This section will cover the different technologies used in the testbed implementation and discuss in detail specific state-of-the-art techniques that are chosen for the diverse characteristics and capabilities to implement energy-neutral WSN. Furthermore, this section studies and analyses WSN nodes, IoT testbeds, wireless research infrastructures and experiment-driven testbeds to set requirements for ultra-low power, industrial WSN testbeds. The developed systems are deployed in a research facility that mimics a warehouse.

3.1.1 Wireless sensor networks

IoT comprises all networked physical systems that communicate opportunistically irrespective of the medium and exchange data. Therefore, this work focuses only on opportunistic, energy-constrained wireless communication by networked nodes in a field or collectively called WSN [12]. The detailed definition for WSN from different articles summarizes that a WSN typically requires little or no infrastructure [12], [13], [14]. They are hosted on the entities in the environment they are deployed in. The deployed devices in a field form a network, where each device is called a sensor node in the network. They obtain data about the environment or any other measurable information in their influence and report the measured properties to a node called a sink node or a data sink. Sensor nodes can work together to monitor and report about the deployed region. The sink nodes are devices that are not energy or resource constrained, whereas other nodes in the network are usually resource-constrained devices [12]. Data can be collectively obtained at the sink devices from all the nodes for further analysis.

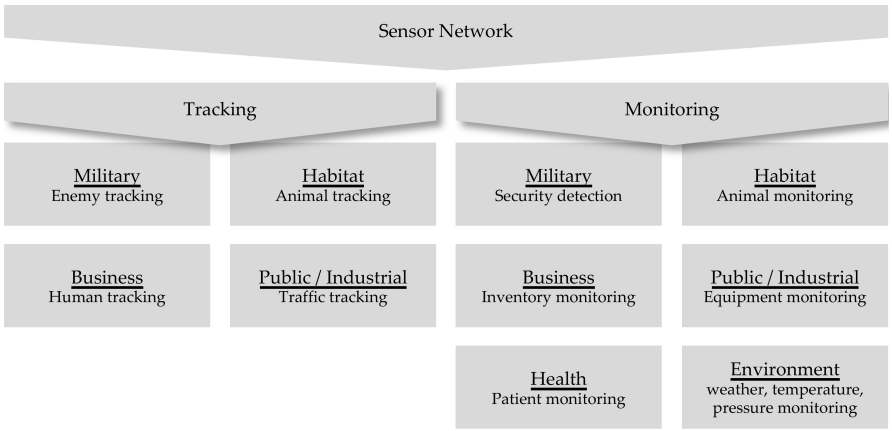

Fig. 3.1 Overview of WSN [12]

WSN can be widely categorized into two types from their deployed nature. The two kinds of WSN categories are Structured and Unstructured [12]. An Unstructured WSN is one that contains a dense collection of sensor nodes deployed in an unplanned manner. Sensor nodes may be deployed in an ad-hoc manner into the field. In ad-hoc deployment, sensor nodes may be randomly placed into the area, or the area may be hard to model for a planned WSN deployment. Once deployed, the network is

left unattended to perform monitoring and reporting functions. In an unstructured WSN, network maintenance, such as managing connectivity and detecting failures, is difficult since many nodes are involved. They are often away from the coverage of the sink nodes. The ideal network topology for unstructured WSN is a mesh network with a multi-hop communication protocol in theory. In a structured WSN, all or some of the sensor nodes are deployed in a pre-planned manner. A structured network's advantage is that fewer nodes can be deployed with lower network maintenance and management cost [12]. Remote sensing applications have a pre-determined system to be monitored and reported. Such systems can use a structured system of WSN where the nodes can be strategically placed with analytical inference. Here the nodes are assumed to be immobile or to traverse a predefined path in a pre-determined manner in the deployed field [12] mostly used in environmental (agriculrutal or building) monitoring. In this work, indoor mobile nodes are deployed to develop an energy-neutral design for WSN.

3.1.2 Applications of WSN

There are diverse fields in which the power of WSN is harnessed to increase the efficiency or productivity of a system. The application of WSN can be classified into two major activities or applications, such as tracking and monitoring. Fig. 3.1 gives an overview of applications of WSN. Fig. 3.1 also shows the two use cases of tracking and monitoring in different fields. For example, WSN deployed in a community for monitoring water consumption could help model and predict water consumption in that community. The data can also provide insights on peak consumption during abnormal periods, leading to the identification of leakage in the water distribution system. Such monitoring could reduce usage by awareness with metrics, model demand from the metrics, and make predictions for the resources to be available [15]. In applications for the public and industrial sectors, fleet management has seen much importance. Deploying WSN for reporting location and acquiring newly calculated route plans has given rise to new business models in warehouse logistics and transport logistics. With this knowledge, intelligent algorithms can be applied to make route calculations efficient or improve the throughput of the fleet by making calculated decisions on centralized data about the fleet, which can also be used in the field of intra-logistics. In building automation, both tracking and monitoring are done. Tracking a room to estimate the number of habitants in the confined space, control the lights, and intelligently reduce power consumption. Monitoring the brightness through the windows of a room to

regulate the light source's brightness in the room is one of the most popular used applications for modern building automation.

3.1.3 Wireless Sensor Network Protocols

A WSN comprises hundreds or thousands of sensor nodes that are densely deployed in an unattended environment [12]. They are capable of sensing, computing, and communicating through wireless media. For such large-scale systems with capabilities for routing, power management, and data dissemination, protocols with specific functions must be designed. The application context also plays a vital role in designing the protocol functions. There are different algorithms and standards through which data is disseminated in the network and outside the system. This section will discuss the design requirements and communication architecture for the wireless sensor networks in detail. This will provide a foundation for designing a custom industrial WSN with an application context.

3.1.4 Communication architecture

There are different scenarios with a massive demand for implementing a wireless sensor network in diverse fields of application [16]. Here we consider an industrial infrastructure with the function of material handling and warehousing. Primarily the different scenarios for WSN communication architecture are discussed in terms of design requirements in section 3.1.5 and then applied to a specific network topology through which the network can be operated reliably. The sensor nodes' primary aim in the sensor network is to make discrete and local measurements about the sensors' phenomena. In industrial WSN, specifically in materials handling facilities, the communication is event-based, with a requirement for high availability of the sensor nodes and the nodes' sensing capabilities [14]. The sensors measure parameters of the atmosphere, machines, or processes to which they are attached to. It also should have capabilities for interfacing with industrial systems to provide actuation capabilities to reduce human intervention on operating the machines. For example, a sensor node will measure radiance, moisture, and temperature in its surroundings and measure the motion of the entity. It is attached in order to infer the actions performed on the entity. Another essential task for the sensor node would be to actuate its entity upon receiving the right message, such as making itself mobile to reach the picking station from the storage racks.

Therefore, the communication architecture of WSN can be generalized to understand the properties that WSN nodes should possess and to develop the design requirements that can be used to design industrially applicable, energy-constrained nodes in PhyNetLab.

3.1.5 Design requirements

Design factors and requirements are essential before deciding on the type of protocol to use for a particular application context. Many researchers have provided guidelines and another important criterion on designing WSN protocols for implementation [17–25], where each of these works discretely provides insights on different design scenarios which were considered depending on their relevance towards this work. A survey of these different terms to frame the critical design requirements for energy-neutral protocols and select low energy wireless sensor network communication strategies is performed. *Reliability* is one of the most vital factors in designing a WSN. It means the system's ability to operate continuously in the failure of one or more components. In the case of the WSN system, the communication standard, hardware, and application resources should be modeled to be fault-tolerant.

Scalability of the networks is the next critical criteria. The network may be implemented in a small industrial infrastructure, which provides positive results. When such an implementation is scaled to hundreds or thousands of nodes, the reliability in the communication within the network might fail, and the nodes' maintenance might become difficult. It is essential to design protocols that are flexible enough to be scaled. Scalability depends on the network's communication, the number of nodes, and the communication reliability between the nodes when the number of nodes increases within a specific area. Therefore, scalability depends on the size of the network and the density of the nodes.

Sensor network *topology* is also considered an essential criterion while designing industrial sensor networks as this defines the placement of the special function devices such as the router nodes and base stations for a given number of nodes. It defines the spatial relationship between the nodes and provides insight into the network's latency, capacity, and robustness. Complex implementations with multi-hop routing criteria provide a large overhead on the packet that arrives at the base station. Therefore, it has to be decided if a router or base station should be placed in a specific location for servicing the sensor nodes or if a multi-hop network is desirable.

Followed by the sensor network topology that influences the costs involved in implementing a WSN are *energy consumption*, and *hardware*

constraints of the nodes. Considering the amount of energy consumption required for WSN nodes, it becomes tedious to replace batteries for maintenance of the nodes in an industrial application context. In a large-scale implementation, the number of nodes replaced with a battery or recharged also influences the cost and reduces the factor of desirability to adapt sensor networks in an industrial process. Energy consumption also influences the reliability of the network. Hardware constraints are imposed on the nodes to improve energy consumption. This reduces the capabilities of the nodes and also reduces the demand for replacing the batteries. The trade-off between reliability and energy can only be decided upon depending on the application. In this work, an energy-neutral design with energy harvesting capabilities is introduced. Therefore, energy consumption and hardware constraints are considered influential factors in developing WSN protocols.

Quality of service is defined by the latency by which the sensor network transmits the data to the base station and the amount of energy it uses to do the same function. The data needs to be transferred within a bound time limit. The power required for transmission must be reduced to a minimum to achieve a better trade-off between the quality of service and energy-neutral design.

Data management is another important criterion when designing WSN. The energy-constrained nodes can only send a fixed amount of data, and they also require data from other nodes that they can use to act on. For such applications, data management and data encoding are essential while designing the protocol. It is possible to design the protocol in a way that data is distributed over heterogeneous networks. Selecting the *transmission media* plays an important role as well as energy constraints, coverage, *network dynamics* and *connectivity*.

In this work, transmission media is confined to 2.4 GHz, 868 MHz, and Ethernet. For an energy-neutral operation, 868 MHz is used. The dynamics in the wireless network are introduce by the factor of mobility and the availability of the nodes. Connectivity is imposed because of the mobility and the density of the nodes in an area. Network dynamics play an important role in resource discovery and intelligent routing. The base stations with more power can transmit at their maximum transmitting power, but the receiving stations that are energy constrained cannot transmit at their maximum power. Therefore, a dynamic and liberating standard for assigning nodes with transmitting power should be designed with the path loss and spatial arrangement of nodes.

Security is essential when sensitive data is being transmitted. A detailed survey of security threats is discussed in the OSI reference in [26]. For industrial applications, any data that reveals the process or the infrastructure's information is considered essential and has to be secured. Some of the security

threats to WSN are described and categorized as passive information gathering, false nodes, node outage, supervision of a node, node malfunction, message corruption, denial of service, and traffic analysis. The networking protocol has to be analyzed for such threats and designed to be robust against these threats.

The criteria mentioned above are considered while designing a WSN communication protocol with an energy-neutral operation. These not only hint at the factors but also provide initial guidelines for developing application-based WSN with an industrial context. Standard network protocols with standardized networking layers defined by the OSI reference model are explored to implement the wireless protocol to operate in energy-neutral conditions.

3.1.6 6LoWPAN

Energy-neutral industrial IoT devices belong to low-power, low data-rate wireless personal area networks [27]. The idea behind such networks is to facilitate low-cost, low-speed, ubiquitous local communication in the proximity across the deployment with little or no demands on the infrastructure. This framework is designed for a communication radius of 10 meters, with a data-rate of 250 Kbit/s. The standard's objective is to create extremely low manufacturing and operation cost with low-power operation [27]. This standard provides real-time suitability features by the reservation of time slots in the communication strategy and reliable communication with collision avoidance through CSMA/CA strategies. AES encryption techniques are implemented at the hardware level. This standard also provides support for power management functions such as LQI, RSSI and energy detection [28], [29]. The devices in these networks are external devices, which perform only the wireless communication or purely embedded, self-functioning devices. Many current MCU units have an embedded radio processor on the same chip to support RF communications. There are many low-power and low data-rate standards that are well established for adaptation in industrial systems. Still, they do not provide support for highly available energy-neutral design on WSN nodes [30]. The standard that defines exactly these conditions is the IEEE 802.15.4 standard [27], [31].

The OSI reference model is used worldwide for standardizing communication protocols [32]. This standard is also defined using the same model where it only describes the MAC sub-layer and the Physical (PHY) layer of the communication strategy. The rest of the layers depend on the application and the limitations of the hardware. Specific applications have

only two nodes connecting in a point-to-point network. In scenarios like these, the network layer and session layer only provide more latency towards the communication topology. Therefore, 802.15.4 defines the MAC and PHY to regulate the medium's shared channel access for many other implementations' coexistence. It leaves the different layers flexible to be adapted according to the application. Even IPv6 for low-power wireless personal area networks can be implemented using this standard. The following are the definitions provided by the IEEE 802.15.4 wireless standard.

3.1.6.1 Physical layer

The physical layer of this standard consists of three prominent RF bands in the unlicensed band. There are other low-power bands as well added to the PHY of 802.15.4 later. The three prominent RF bands are 1) 868 MHz, 2) 915 MHz, and 3) 2450 MHz (2.4 GHz). In this work, the two bands, 868 MHz and 2.4 GHz are used [31]. In this standard, there are different types of DSSS modulation techniques used. BPSK and O-QPSK are the primary types of modulators for 868/915 and 2.4 GHz modulation. Other modulation techniques are also used in the PHY of 802.15.4 [30]. They are FSK, GFSK, and OOK. The 2.4 GHz is implemented using ZigBee network protocol with its standard PHY as defined by 802.15.4 standard in the PhyNetLab implementation, whereas 868 MHz is used for implementing energy-neutral communication, which takes advantage of all the low-power capabilities in designing the PHY.

3.1.6.2 MAC layer

The MAC strategy is defined by this standard for managing the access of the shared physical channel for the nodes by guaranteeing time slots. Network beaconing is another function performed by the MAC layer. Network beaconing is a management frame that is transmitted periodically to announce the presence of a wireless network. It is transmitted by the access point in an infrastructure to announce the network's state for other nodes to synchronize. The beacon frame consists of the timestamp, which allows all the other receiving stations to synchronize their clocks with the access point. The beacon frame is transmitted periodically. A time unit called the TBTT is transmitted with a beacon frame for the stations in the network to announce beacon frames' periodicity. Beacon frame also carries depending on the type of the PHY layer, a frequency-hopping parameter set or direct-sequence parameter set, a SSID, supported data-rate, and a traffic indication map, and

a contention-free parameter set [30]. The MAC layer also controls frame validation, guarantees time slots, and handles node associations. 6LoWPAN is adapted for IEEE 802.15.4 standard depending on the application [30]. When specific standards like IPv6 or IPv4 are adapted, the standard frame size is more than 127 bytes. The MAC layer is supplemented with adaptation layer protocols, which provide fragmentation of the frames and header compression to support large packet sizes required by those standards. The above mentioned are IEEE 802.15.4 standard definitions of low-power, low data-rate, wireless personal area networks.

3.2 PhyNetLab: Ultra-low-power WSN testbed

In this section, PhyNetLab's architecture design is discussed with the various features deployed in the infrastructure. Initially, many Internet testbeds, networking infrastructures, and wireless measurement testbeds were studied. Performing distributed measurements in WSNs is a critical feature for PhyNetLab. The derived IoT testbed requirements from the testbed survey in A.0.5 and the architecture design are made so that the system could be replicated to any industrial application. The testbed architecture is consolidated to provide a reference model for an energy-constrained, distributed infrastructure for deploying wireless sensor networks in an application-based industrial system.

3.2.1 Requirements for an IoT testbed

For any system, its architecture defines the operation with better qualitative aspects desired for industrial adaptation [11]. In this section, the primary qualitative aspects of industrial testbeds are discussed. These aspects are considered the requirements of an industrial WSN deployment and are incorporated during the design phase of PhyNetLab. PhyNetLab is an industrial testbed because it is implemented in an industrial material handling facility. Experiments held in the testbed can relate to either communicating an industrial facility's dynamics or simulating a business application to estimate performance. The PhyNetLab system architecture takes the findings from the surveyed testbeds as design considerations. There have been platforms for experimentation and research with state-of-the-art outcomes for WSN techniques concerning communication strategies in low-power platforms. In this testbed, the energy-neutral operation of the nodes is focused on the logistics facility application context.

One of the goals of the PhyNetLab testbed is to improve the rate of industry adaptation of WSNs.

Various testbed architectures are studied to narrow down the design specifications for PhyNetLab. These are considered the pillars for developing a robust WSN testbed with a context for energy-neutral operation in industrial applications. It is highly improbable to simulate an industry environment to different model modalities. The dynamics of industry operations require performance availability scenarios, which become more complicated to simulate [14]. This provides the motivation to implement the testbed in the materials handling facility [11]. The next requirement for the testbed is the experimentation experience. It becomes tedious when there are 350 nodes in the sensor-field that communicate with six access points. A distributed architecture with the real-time operation is proposed for the PhyNetLab implementation to provide a multi-user experimentation context for the data and the nodes that generate the data. A management portal is built with an information technology infrastructure to accommodate a multi-user context with less human intervention to improve experimentation.

Energy neutral operation and energy harvesting are two sides of a coin. For research on an energy-neutral design on WSN nodes, studies about energy harvesting should be made [11]. Energy harvesting requires an environment close to a real case to study the factors influencing energy harvesting limitations and to implement an energy-neutral design in smart industrial objects. Seamless experimentation experience is required for quickly testing iterations over operational concepts and determining the nodes' performance availability. These types of experimentation can be performed remotely over the Internet using a management tool [11].

Scalability is the capability of a system to handle growth in the number of nodes in the WSN installation. In scenarios like those, the system should be modeled to accommodate the growth either by increasing the potential to handle the system's development or by making multiple instances in a decentralized manner to reduce the load on a specific system [11]. There are two types of scalability, horizontal and vertical. In the testbed context, horizontal scalability should be addressed, and the system should be designed to scale with the demand for growth in the nodes. Scalability is considered in the design of PhyNetLab as one of the factors due to the dynamics involved in industry operation standards. It is essential that these systems are highly scalable and that their availability could be increased by increasing the number of nodes without any outage in the network. The database systems to store the data in an indexed manner are also chosen so that the whole system is robust against issues arising from scalability [11]. The architecture supports plug and play of devices when pre-installed with the distributed software implemented across nodes. Scalability is possible

on both dimensions within the PhyNetLab architecture [11]. When there is horizontal scaling of nodes, i.e., the number of nodes increases, there should be vertical scaling of the access-points and database systems, i.e., to improve the resources for such systems. Therefore, the testbed architecture is designed in such a way that increasing horizontal scaling inherently scales vertically using the in-memory data grid.

The heterogeneity of devices is considered to improve the interoperation of different existing tools in the industry. As cost is a factor for industrial operation, it is impossible to equip the whole system to adapt to the same hardware platform, rather the system is modeled to have interoperability. The PhyNode itself provides heterogeneity concerning the PHY medium used. There is support for 2.4 GHz, and sub-1 GHz band in one node and interoperability of these nodes can be realized using the PhyNetLab management platform [10]. The interoperability of the two systems is targeted by implementing shared data storage. This unites the two data driven WSN systems at the data storage. This approach reduces complexity in implementing a unification stack for two communication stacks.

The six principal architectural requirements for a WSN testbed are

1. Experimentation experience
2. Environment
3. Availability
4. Scalability
5. Heterogeneity
6. Industrial impact

Impact here means adaptations of WSN in the industry, which is the motivation for the testbed. Replicating the environment for better testing conditions with seamless experimentation tools, providing heterogeneity in the device nature to incorporate existing installations, and providing real-time access to data (availability) reliably are considered vital elements for the impact of such a WSN testbed. The testbed implementation also hints at the cost required for such an implementation. Energy-neutral design requirements for indoor photo-voltaic energy-harvesting systems can be defined. Henceforth, the architecture design of the testbed revolves mainly around six principal requirements as listed above.

3.2.2 PhyNetLab architecture

PhyNetLab is designed as an ultra-low-power WSN testbed with the design requirements, as discussed earlier. The testbed architecture considers the six principles, as mentioned earlier, as requirements to model the testbed for

experimenting on the end nodes' energy-neutral operation with an application context. A generalized approach towards the testbed architecture is taken to model a system; this can work in a modular configuration where two of the architecture layers can be implemented as a standalone system. Depending on the application's requirements, a layer can be omitted from implementation or scaled horizontally across locations [11]. The modularity in the design of the architecture and the state-of-the-art techniques from distributed computing enables such a configuration. The architecture is a decentralized, real-time distributed architecture for ultra-low power application-based WSNs, where the application defines the requirements. The testbed architecture is defined as a three-tier architecture with a prospect for high scalability of nodes and systems [11]. Another essential factor of distributed systems is the availability of data across tiers of the application and locations of the infrastructure [11]. Design considerations for scalability, availability, and reliability of the testbed, among other factors, significantly account for industrial adaption's desirability. After performing the architecture design and vertical integration of the system, an industrial application is designed with the design guidelines provided by PhyNetLab [11]. One of the design assumptions inspired by the industrial requirement, which also impacts the operation of WSN applications, is that all field nodes in the testbed are mobile. The field nodes are called PhyNode, and they are attached to the physical entities in a materials handling facility. The mobility of the nodes in PhyNetLab is achieved by a fleet of an Autonomous Ground / Guided Vehicle (AGV), as shown in Fig. 3.2. The sensor nodes are attached to the crates in a materials handling facility. PhyNode and other hardware systems that are deployed in the architecture are described in chapter 3.3.

Fig. 3.2 The logistics research facility where PhyNetLab was planned with a source for outdoor lighting and AGVs for mobility.

Fig. 3.2 is a photo-realistic rendering of the industrial facility where PhyNetLab is deployed. The testbed has three tiers, with the topmost level (Tier I) consisting of enterprise-grade cloud servers that have the power of performing complex computations. The servers also interact with remote users, store valuable data from the sensor networks, and manage the systems by delivering firmware from the user using the internet [11]. This layer implements the three functions of the PhyNetLab testbed. It has three servers: the application server, database server, and the build server. The parts of these servers, which implement different functional aspects of PhyNetLab, are encompassed within an API server, as shown in Fig. 3.3 for data exchange of the various servers with the entities outside the PhyNetLab implementation. There can be another location of industrial WSN with heterogeneous devices and a communication stack or a third-party application that consumes the data from the WSN testbed.

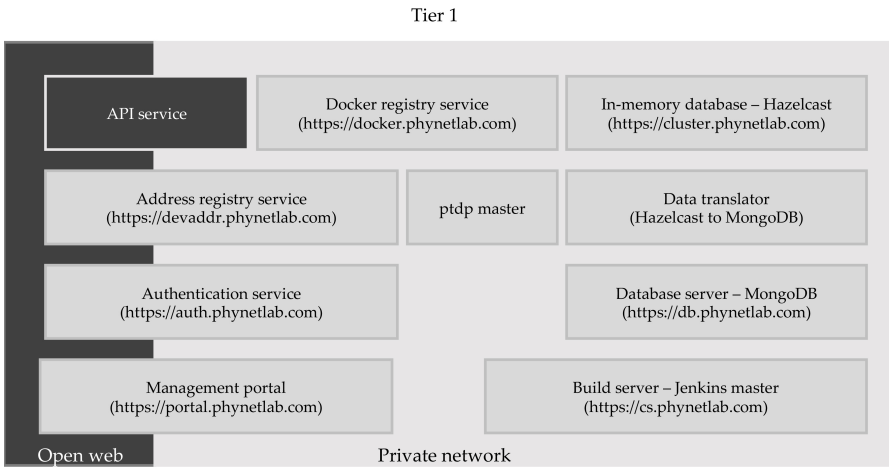

Fig. 3.3 Three categories of servers deployed on tier 1 of PhyNetLab

The three-tier architecture of the PhyNetLab is illustrated in Fig. 3.5, where the second tier of the architecture has edge devices of the IoT architecture. Edge devices are located at the sensor-field where the WSN nodes are deployed. These devices bind the higher-level application servers that consume data in a business context with the sensor nodes and controller devices on the WSN platforms deployed in the sensor-field. These devices are called access-points in the context of PhyNetLab. There are six of those access points deployed in the materials handling facility with capabilities for communicating with all the sensor nodes and the servers on tier 1. They are deployed in a distributed manner i.e. the data from the

sensor node is received at an access point in PhyNetLab. The received data is distributed in real-time to all the access points to handle the nodes' mobility and the sensor nodes' dynamic associations with the access point. A NodeJS based event logger is deployed in the access point, which improves debugging possibilities of each access point. Fig. 3.4 illustrates the functional block diagram of a tier 2 device deployed in PhyNetLab.

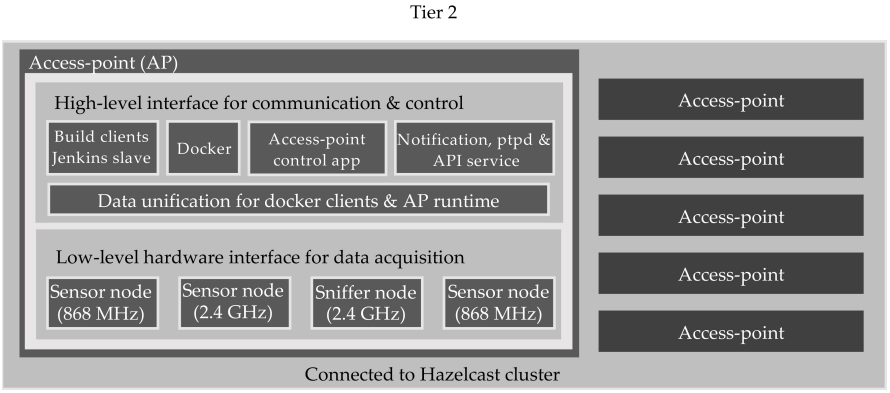

Fig. 3.4 Functional block diagram of a tier 2 device

Tier 3 is the actual focus of future research, which is deployed in the material handling facility and consists of 350 PhyNodes [11]. The PhyNodes are connected to the user for remote interaction using the ZigBee back-bone network. This architecture compiles all the principles that were discussed earlier for the architecture design of the testbed. There is a dynamic data flow over the network with user authentication. Each tier is vertically scalable, or horizontal scaling of the levels across locations to form clustered operation is also possible. Fig. 3.5 shows the consolidated architecture of all the three tiers with the extension of their features over the different levels of PhyNetLab. Fig. 3.5 shows that the identification service runs through all the service layers. This identification is available in all the data platforms by either accessing an API service for systems outside the PhyNetLab implementation or using the in-memory data grid identification service. The systems and nodes associated with the testbed, particularly the nodes in the same location, can access the in-memory data grid. All other systems and nodes can only access the API service with proper authentication. The identification service must be modeled to be a highly available system. It will identify the testbed devices and assign new devices to the testbed when they are registered to the testbed platform. This improves the robustness of the testbed against device spoofing. Device spoofing happens when nodes in the testbed retain an invalid address assigned to it during a previous

experiment. When this address is assigned to another device, two devices receive messages as they are assigned the same address. To tackle such issues, an identification system is designed to update the device address centrally with unique identification strings that are used by the end nodes upon rejoining with the network after abnormally long sleep periods. A time-limited token system like the JWT is used where the token should be renewed after the timeout. If expired, the node will receive a new address and, if required, an identification string from the identification service.

An authentication service is also available across the testbed to allow authenticated users to access the data and the end nodes. In PhyNetLab architecture, LDAP is used along with PassportJS for the management portal to provide a single database for user authentication across the testbed. The build server also spans its service throughout the testbed, where it is used for securely iterating the software components of PhyNetLab. It is used for building software for an access-point, management portal, and authentication and identification service. It is also planned to manage Texas Instruments Code Composer Studio to compile code for the PhyNode research platform. This will automate how the software for the PhyNode is built and distributed across heterogeneous networks. The firmware is pushed to the access point to which the node is associated and transferred to the destination platform using the ZigBee back-bone network.

3.2.3 Distributed real-time architecture

PhyNetLab architecture works in real-time, i.e., the time synchronization is taken care of by a time server using the Ethernet network. An application is defined to be working in sync if two or more devices communicating in a network are synchronized to the same clock where the offset of the time between the two nodes is at a minimum. They comply with a standard time unit within which the operation between the two networked devices is completed [33, 34]. One option to achieve that level of time synchronization between more than two hardware devices is to synchronize each of the devices to the clock by the GPS signal. It is expensive to implement external hardware to receive the GPS signal for each of the access-points. Moreover, for indoor devices, signal acquisition is low. Therefore, a software-implemented time synchronization protocol, called the precision time protocol, is used in this implementation. Its quality of service claims to be within the bounds of 10 microseconds between the secondary and the master [35]. The precision time protocol is used across devices in the first two tiers from the top, as shown in Fig. 3.5. The master is a virtualized server running Ubuntu 14.04 on the topmost tier with secondaries on the

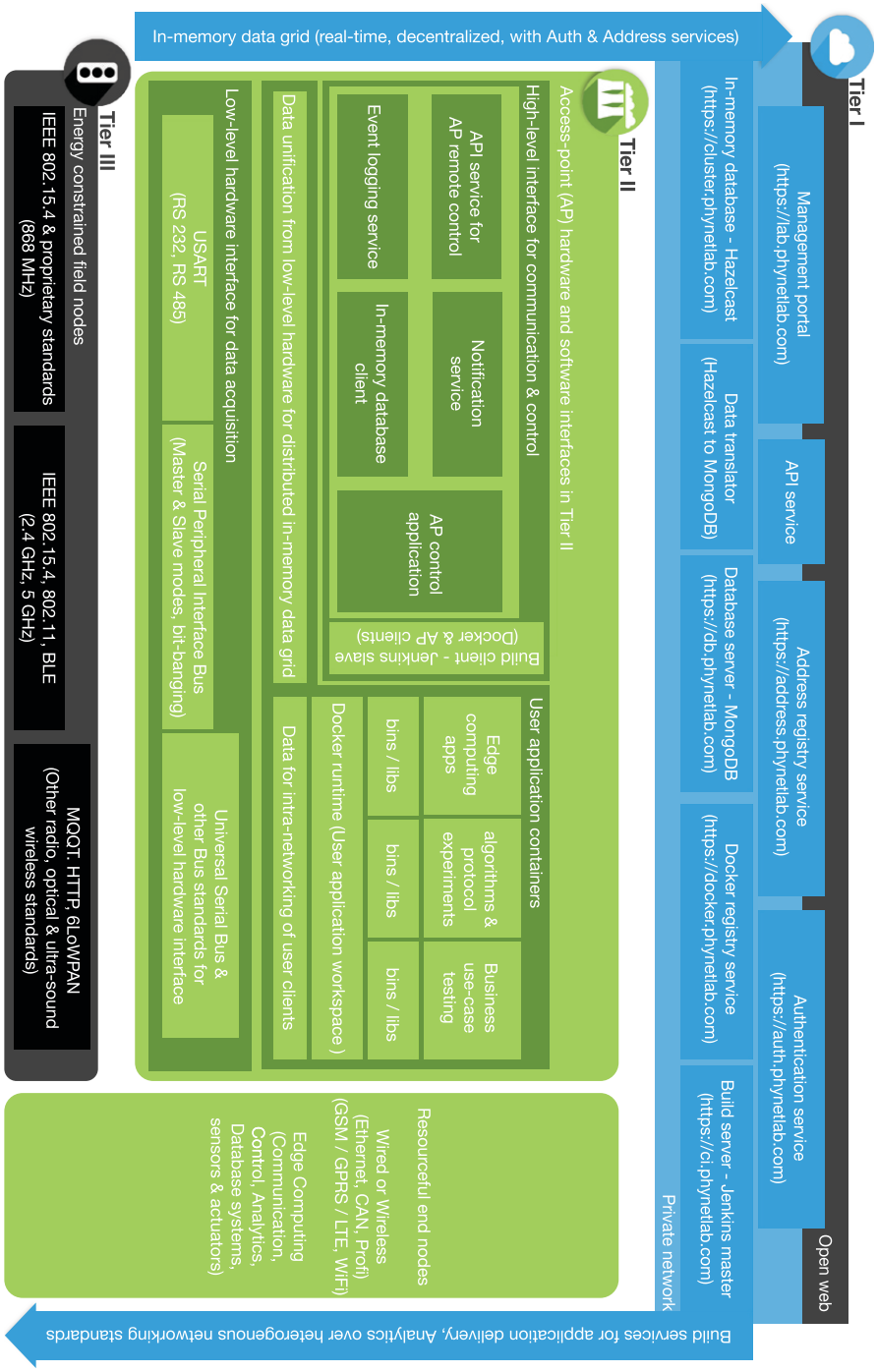

Fig. 3.5 Consolidated centralized testbed reference architecture

testbed's top and middle tiers. This means all the access points are synchronized with the database's clocks, application, and build servers. It is desirable to run the timeserver and the devices in the second tier to reduce network latency.

For devices to have a precise time and when the server does not have internet access, there are possibilities such as a grandmaster clock which synchronizes itself with the GPS signal and provides the correct time to the server. The devices don't have to have the same time as the coordinated global time but synchronize the network's time. The time is synchronized to the global clock on the time server with the Internet's network time protocol application. An experiment was conducted to define the precision time protocol limits to be implemented in the testbed. There were five secondaries on Tier II for this experiment, and the master clock was in Tier I with other servers. The master and the secondaries were in the same subnet running Precision Time Protocol Daemon (ptpd). It is available for installation directly from the repositories. It can be installed with the following one-line terminal command executed with root privileges, as shown in the first line of the listing 3.1.

Listing 3.1 Setup procedure for ptpd in master and secondary nodes [36]

```
$ sudo apt-get update & sudo apt-get install ptpd
$ sudo ptpd2 -M -i eth1 -f ~/Desktop/ptpd21.log
$ sudo ptpd2 -s -i eth0 -f /ptpd31.log
```

The initial setup for the ptpd is to assign one of the nodes to run as a master and the other nodes as secondaries while starting the service in the terminal window with these command line arguments. The command line argument for assigning a master is with -M and for a Secondary is -s. With the initial setup, the log file path is also given using the argument -f, followed by the file system's location. The second and third lines in the listing 3.1 are the examples for starting the ptpd daemon on the master and secondary, respectively. This log file is used for later analysis to estimate the mean-offset between the secondary nodes to prove if the nodes' clocks are within a desired range of operation. Optionally during the initial setup, the network interface to be used for packet broadcast by the master and interface to listen by the secondary can be assigned with the argument -i, followed by the network interface's name. This helps the ptpd program choose the right network interface if more than one interface is activated. From the log files, offsets from the master with their timestamps are extracted for each log file.

Fig. 3.6 shows the offset from the master plotted against the timestamp. It can be seen that the nodes have the same signature as each other. This means that they have a regular offset from the master. All the secondaries provide the same results for an experiment period of 24 hours. This offset from the master is calculated for all the secondaries. The arithmetic mean of

these values provides an operating range for the secondary nodes. Even though there is an offset between the secondary and the master, the offset signature is almost equal. Thus five secondary nodes provide an offset footprint that is similar to each other with an upper limit of 300 μs from the master. The time synchronization takes up to five minutes, after which the difference is stabilized between the master and secondary. This proves that

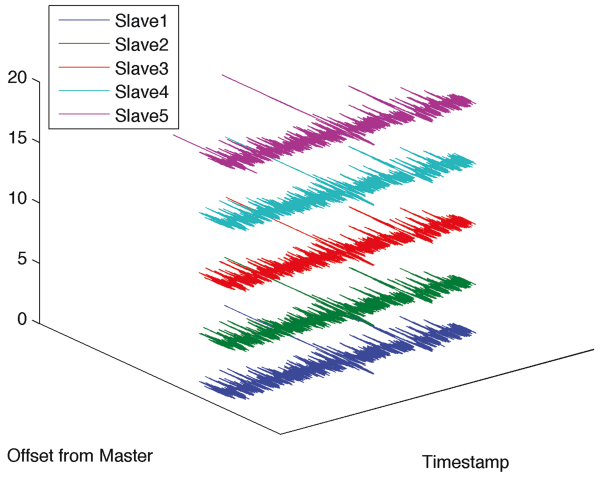

Fig. 3.6 Time offset of each secondary node from the master

the implementation of a precision time protocol can improve the synchronization of the nodes in the network, providing a better timestamp for data arriving in the network from the sensor nodes in the field. It is useful for receiving the same data packets but with different timestamps because of precision. The time of flight of the packets can be calculated with higher accuracy. All other servers in the network implemented with PhyNetLab acquire time from this master clock time server.

3.3 PhyNetLab Hardware Systems

PhyNetLab uses a wide variety of hardware in its testbed ranging from energy-neutral field nodes to cloud servers with on-demand access to resources at scale. The energy-neutral devices are one of the field nodes deployed in the sensor-field in the materials handling facility. The testbed is designed to host heterogeneous devices on any tier of the testbed

architecture. The testbed implements a border device that connects to the Internet, which translates the devices' IP to the device address in the field. Other devices are connected using the Ethernet interface to the servers either in the second or third tier using web services. The devices in the testbed architecture's various tiers are described in detail with their block diagrams as shown in Fig. 3.12 showing their hardware components and other devices using various services. The hardware in PhyNetLab can be widely categorized into three device types 1) field nodes or PhyNode, 2) Router or access-points, and 3) Application servers or remotely collocated cloud servers.

3.3.1 PhyNode

The PhyNode hardware is intended to create a platform to test various wireless protocols on the sub-1GHz band and 2.4 GHz band for varying use cases of the Industry 4.0 standards. Individualized production, horizontal integration in collaborative networks, and end-to-end digital integration are some of the applications in Industry 4.0, as described by Brettel, Malte, et al [37] [11]. PhyNode is used in testing neutral energy strategies with the operating software on the processing unit or the communication strategy with the radio unit and the supporting processing unit. Implementing in materials handling facility or a mimicking facility logistics environment to test the WSN nodes can overcome any simulation limitations. PhyNode is in the third tier of the testbed architecture. It uses custom software, written in C language, to implement user configurations in testing communication strategies and optimize the processing unit's operation to operate in energy-neutral standards.

3.3.1.1 System design description

This section elaborates on the PhyNode's general configuration, system design, and hardware architecture. The electronic components used in PhyNode to provide features for the application and achieve energy-neutral operation are detailed in the subsections. As mentioned in the comparison, the hardware platform has two radio transceivers. The configuration is made in such a way that there are two nodes on the same platform. One of the platforms uses different RF channels than the other. This enables the two nodes, in such proximity, to coexist. The objective of having two nodes on the same platform to provide diverse RF bands is to experiment with an industrial application. The node part with CC2530 is battery powered and

can be used as a platform that complements the experimentation experience by reporting and monitoring the health of the slave node. The slave node is the CC1200, which has an energy-neutral operation and can be programmed using the master board of the CC2530 part board. The MCU core of CC2530 is an 8051 architecture. In contrast, the CC1200 is a radio front-end chip controlled by an MSP430 with 16-bit Reduced Instruction Set Computing (RISC) Microprocessor without Interlocked Pipeline Stages (MIPS) architecture.

The MNB is the central hardware platform that forms a controller network for a SSB radio chip held by the MNB. A master network is a backbone network that provides insight into the communication and energy characteristics of the SSB. The controller network is operated at the 2.4 GHz band. The operating software of MNB is ZigBee. Applications addressing other software systems such as TinyOS or ContikiOS, are compatible with this system MCU architecture. These operating systems are tested with more than one node for performance, range, and packet error in a lab setup. In general, the IEEE 802.15.4 protocol standard can be implemented with any operating system on the 8051 MCU core. The main objective of having such a network is to collect metrics about the SSB and deliver Over-the-Air (OTA) firmware updates for the slave board that make sensor nodes highly mobile in the deployed site. Therefore, it is easier when there is a system that can facilitate flashing the software process remotely.

Fig. 3.7 shows the 8-pole connection and various other essential components of PhyNode explained in detail. From Fig. 3.7, it can be seen that the SSB can be easily removed or changed with the help of holders that fasten the SSB with the MNB [10]. This is a feature implemented to iterate hardware quickly. There are possibilities for the MNB or SSB to be damaged beyond operation due to a user program or other environmental characteristics like physical damage or electrostatic discharge. It is easier to swap either one of the damaged boards with new hardware. It reduces the cost of the implementation since the hardware used in research is bound to be damaged.

3.3.1.2 System Architecture MNB

MNB provides the skeletal framework of the PhyNode. Its processor core is powered with a CC2530 IC from Texas Instruments (TI). Fig. 3.8 shows the MNB system architecture with its functional components. A System on Chip (SoC) integrates functional components of an electronic system into a single chip. The CC2530 SoC contains an 8051-core MCU with five-channel direct memory access and a radio transceiver supporting the IEEE 802.15.4 standard. Easy pin mappings provide interfaces to the chip, such as UART,

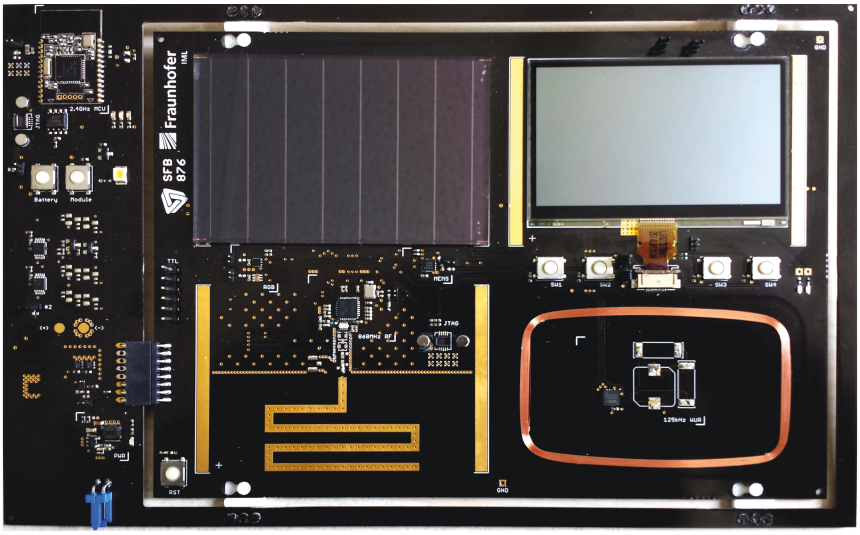

Fig. 3.7 PCB layout of PhyNode with MNB and SSB (inside the cut out) [10]

SPI [29]. The MCU core is an integrated low-power 8051 MCU architecture [29]. The selection of such an RF SoC is mainly due to its low-power operation and a small spatial footprint in the PCB. This module supports a PCB antenna for the 2.4 GHz band operation. The daughter boards from the CC2530 have a booster antenna module, which provides a better antenna gain. However, the trade-off provides less spatial footprint on the PCB with an allowable loss in the antenna gain. It is considered an acceptable trade-off for the features it provides to PhyNode. A Li-Polymer Battery with 2000 mAh powers the MNB that provides sufficient energy for long-term operation with the low-power SoC, powering the SSB when required and providing enough stable power for flashing a new firmware to the SSB. For battery protection and the battery's flexible operation, a power management IC is added to the module [38]. BQ29700 is the chip used for flexible power management [39].

An additional feature of the MNB is a high-power infrared (IR) LED. It is planned to provide a testbed feature that will help verify novel indoor localization algorithms. A camera that can detect node locations can be used to estimate the nodes' absolute location in the sensor-field. An IR LED can be controlled using the GPIO of the MCU in SSB that facilitates a method to identify each node uniquely with the frequency with which they are turned on and off.

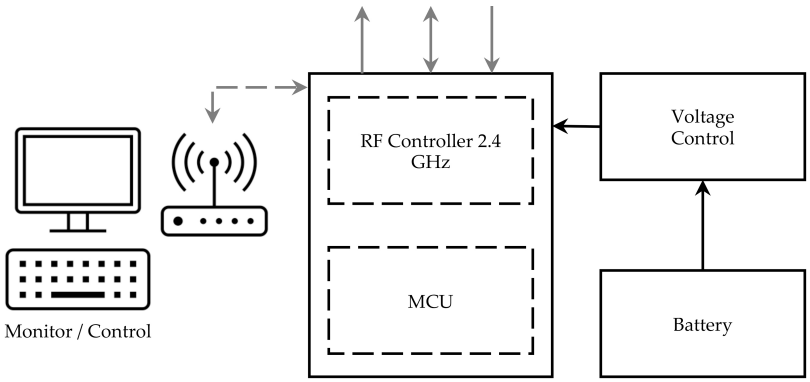

Fig. 3.8 Block diagram of the MNB [10]

3.3.1.3 System Architecture of SSB

SSB is the energy-neutral node by design, and its block diagram is illustrated in Fig. 3.9. It has photo-voltaic harvesting capabilities and an ultra-low-power radio transceiver that operates in the sub-1 GHz band. On the heart of the SSB is an MSP430FR5969 MCU manufactured by Texas Instruments. It comes from the family of low-power MCU with a 16-bit RISC architecture core. It has a FRAM-based on MCU with power consumption from 0.02 μA in sleep mode up to a 100μA in an active state. FRAM is a groundbreaking technology for MCU, which increases the speed at which the controller reads and writes to the RAM. The speed is considerably better than other RAM technologies and reduces each read and write data transaction's energy requirement. As a whole, reduced energy consumption and increased access speeds reduce the wake time during a memory access, which results in the total energy consumed during memory access. This has a positive effect on the operation of nodes with an energy-neutral design. Compared to a conventional flash RAM, the 64kb FRAM in the MCU is very flexible concerning memory access and is highly energy-efficient.

3.3.1.4 Radio front-end

For RF communication, the TI CC1200, a sub-1 GHz, ultra-low-power transceiver, is used on the board [28]. The MCU communicates with the transceiver using the SPI bus. The transceiver has three programmable GPIO, which are used to provide wake-up events and other notifications for the controller's programming logic. One GPIO can be programmed to

provide an interrupt event upon successfully receiving a packet with a valid Cyclic Redundancy Check (CRC). The chip provides CRC checking in the hardware itself, which reduces the computation energy required to check for every packet at the MCU [40]. Another GPIO can be programmed to invoke an event at the MCU when it does a CCA. The channel's spectral energy is measured, and when it is less than a threshold, the channel is considered free, i.e., no other stations are accessing the channel. This is used in programming the MCU to sleep for definite cycles before sensing the channel for CCA in the next slot. Other programmable outputs can be configured for the GPIO, such as notifying the transceiver state, the end of transmission, an interrupt event when a packet was received with an overflow in the RX FIFO buffer [40]. An additional feature in the SSB is to add a 125 kHz wake-up receiver to reduce the nodes' listening time. It is implemented via an inductive coupling coil as a receiver antenna on-board. It can react to a specified pattern sent on this channel. This signal with a pattern works as a wake-up signal for the MCU that enables the MCU to switch between sleep mode and active mode.

3.3.1.5 On-board sensors

Five onboard sensors are used for measuring and monitoring the environment of the PhyNode. Ambient sensors are essential for a hardware platform with energy harvesting capabilities [41]. There are two MAXIM ambient sensors implemented with features to measure RGB color, infrared and ambient light. This sensor's primary function is generating a map with potential optical energy harvesting. Predicting expected power from the photo-voltaic cell is important for ensuring a reliable operation of energy neutral devices [42], [43]. Two temperature sensors are used, including the MCU and one included in the MAXIM ambient sensor. Temperature is a critical measure since the photo-voltaic cells also depend on an optimum operating range for their energy production [41]. Furthermore, a 3- axis accelerometer is also available.

SSB energy-neutral design operation means that the SSB's energy source is supplied by an energy harvesting module for active communication and limited computation on the SSB processor core. Maximum power point (MPP) tracking is a highly accepted technique for efficient energy harvesting using photo-voltaic cells. Getting the maximum power possible from photo-voltaic modules by changing the physical or electronic configuration is one of the properties. For this purpose and for battery management, an ultra-low-power harvester IC with boost charger and autonomous power multiplexer is used [44]. It is designed so that there is a possibility to switch to an alternative energy source if the photo-voltaic

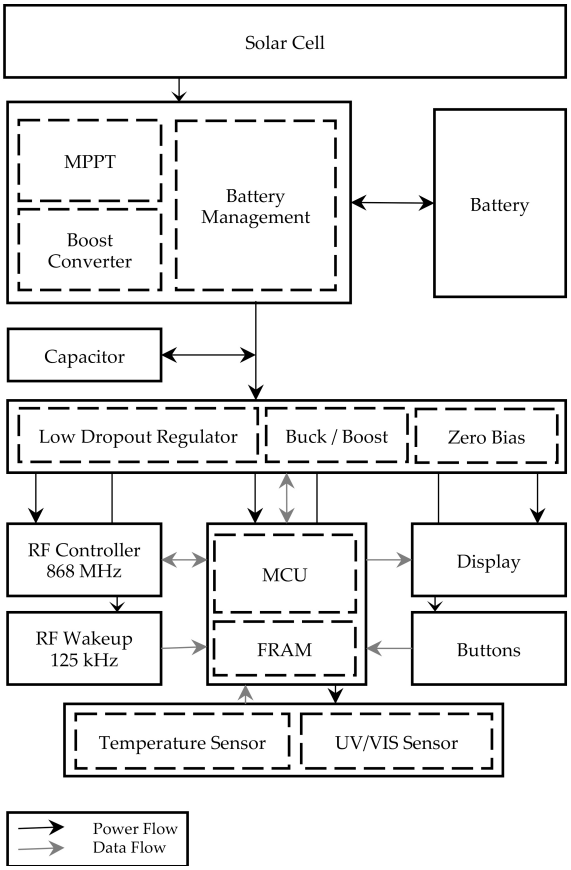

Fig. 3.9 Swappable secondary board system architecture [10]

charged energy storage goes empty. Such a measure is taken to make the hardware platform robust in terms of power and efficiently understand and model its power requirements. Fig. 3.9 shows all the components of the SSB as a block diagram with the data and power flow between its parts.

3.3.1.6 Energy characteristics of SSB

PhyNode has two parts with different radio front-ends, one has an energy-neutral design, and the other has a low-power operation with a different operating RF. PhyNode has a photo-voltaic cell (7072048 Solem Cell) on the part designed with the energy-neutral operation. It is part of the SSB. The motivation is to provide a longer operating duration than usual, compared to low-power wireless sensor nodes without intervention for

recharging or replacing the battery. This is designed to be at a low cost and targeted for industrial adaption because of its features. It has an energy-harvesting module that can provide insight into the capacity of indoor energy harvesting and model computation and the sensor node's communication capabilities to operate in an energy-neutral manner. Energy modeling for the photo-voltaic panel was studied using the appropriate photo-voltaic panel in PhyNode, which is detailed in [41]. The results of a comparative analysis of the power produced by the selected photo-voltaic cell in indoor lighting conditions are modeled and described in [41]. The energy produced is modeled in [41] as the function of indoor artificial lights and reflective sunlight during the day.

3.3.1.7 PhyNode intra-connection

A connection is required to take control of the SSB. The connector can do this with pin extensions. The configuration of the pins for the connection between the boards is shown in Fig. 3.10. RX and TX are used for communication between MNB and SSB. It is a full-duplex communication line between the two boards. The Bootstrap Loader is firmware from the manufacturer that is used to flash the MCU. It requires two pins for which reset (!RST) and TEST pins can be used. The master board or MNB can enter the Bootstrap Loader (BSL) of the SSB to flash a new firmware provided through the ZigBee application layer. The MCU is the reference voltage from the SSB MCU unit used as a reference for an alternate power source. VBAT can be used as an

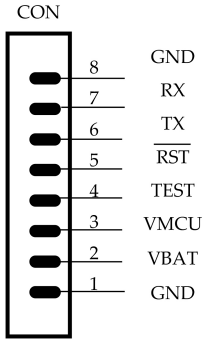

Fig. 3.10 8-Pole connection between MNB and SSB [10]

alternative power source. This is designed to provide an external alternative power source if the energy harvester's storage battery is empty due to over-exploitation of the hardware resources. Such a configuration reduces the

amount of time for a person to flash new software onto the SSB. Calculating economically, the amount of time spent on reprogramming the nodes for each iteration during the experimentation phase renders the testbed useless [10]. The MNB is also a hardware platform that could be used for researching WSNs with a 2.4 GHz band apart from its use as a controller network. The disadvantage is that the energy-neutral design does not extend also to the MNB. The SSB, connected to the MNB, has a TI CC1200 RF transceiver that works in the sub-1 GHz band. This part of the research platform fits energy-neutral functioning by design [10].

3.3.2 Access-point

In this section, the hardware of the access-point is detailed with its interface to the low-level hardware. The different communication strategies enabled with the access-point as a result of the low-level hardware are described with a functional block diagram and its software stack.

3.3.2.1 Hardware specification

The access-point is deployed in the network as a central hub in a star topology. The hardware is enclosed in an industrial casing as it is deployed in an industrial facility for experimentation. The hardware consists of a low-power computer that can interface with low-level hardware such as the radio modules and high-level hardware to connect to the internet. The hardware that is chosen for the implementation is a RPi. It provides low-cost computer power with an ARM processor at its core. It has 40 pin headers for low-level interfacing hardware that can be directly programmed with the data they generate [45]. The hardware, collectively with the 868 MHz and 2.4 GHz radio front-ends, can be called an access point. The hardware components in the PhyNetLab router hardware are two CC1200 daughter boards, two CC2531 USB sticks, power management for the RPi, and the connected modules. A tricolor LED strip to improve notifications from the hardware and an Ethernet cable that provides internet connection. The interconnection between other sensor field access points can be achieved by assigning all access points to the same IP network. In the future, it is planned to implement the transmitted 125 kHz wake-up signal on the access point to improve the availability of the long sleeping nodes. This should be interfaced with a software stack of access-points to provide connections for each node's physical node and signal pattern.

3.3.2.2 Low-level hardware interface

Four modules are connected directly to the RPi pin headers. There are two CC1200 daughter boards connected using the SPI interface. Their three programmable GPIO are also connected to the PIN headers to program logic that can improve the interconnected hardware layers' operation efficiency. The other two modules are CC2531 USB sticks [46]. There is a facility to connect them via the headers, but it is also possible to directly connect them onto the USB hub on the RPi. The CC2531 modules do not have a better antenna gain compared to the CC2530 daughterboards as they provide a booster antenna circuit to improve the reception gain. It is planned to connect the CC2530 modules onto the RPi using the GPIO pin headers over the USART wired communication standard to improve air (channel usage) efficiency.

3.3.2.3 Communication strategies

The RPi has more than one standard for communicating as it is deployed in the middle layer of the three-tier architecture. The middle layer is a coordination layer which reduces the computation load on the end nodes to supplement better energy efficiency and as a translator that accesses the application servers to provide field data for the nodes from the application data. The access point is connected to the Internet by the TCP/IP protocol stack over the wired Ethernet interface. This provides an interconnection between the access-points that are implemented in a facility. Here, state-of-the-art distributed computing techniques reduce data latency between the access-points and improve coordinated operation between collocated access-points. There are four USB connections possible on the RPi, of which two are free and can be used for planning future operational features.

The RPi communicates in the 2.4 GHz band, implementing a ZigBee network in the testbed as a backbone content delivery network for the PhyNode platform. It is achieved by connecting an SoC to the RPi via USB in the implemented version. The SoC implements the ZigBee communication strategy for communicating with the field nodes. RPi provides Internet interfacing with other nodes over the Internet and offers better computational capabilities. The application endpoint of ZigBee is available on the SoC with a connection using the USB, where the SoC is connected for power and data exchange. Initially, the USB connection is utilized with the CC2531 USB sticks that provide a virtual comport connection over USB [46]. The future version is planned to connect the CC2530 daughter boards directly over the GPIO headers. This is a drop-in

replacement for the USB sticks for a better reception gain as it is also interfaced using USART communication over the GPIO headers. There are two CC2531 connected to the RPi; one of the CC2531 is used for connecting with the ZigBee network. The other CC2531 module is used as a network sniffer to record every air message from the field nodes. The sniffer module uses a TI-supplied firmware, which connects to the RPi using the USB protocol to improve the data-rate between the module and the RPi.

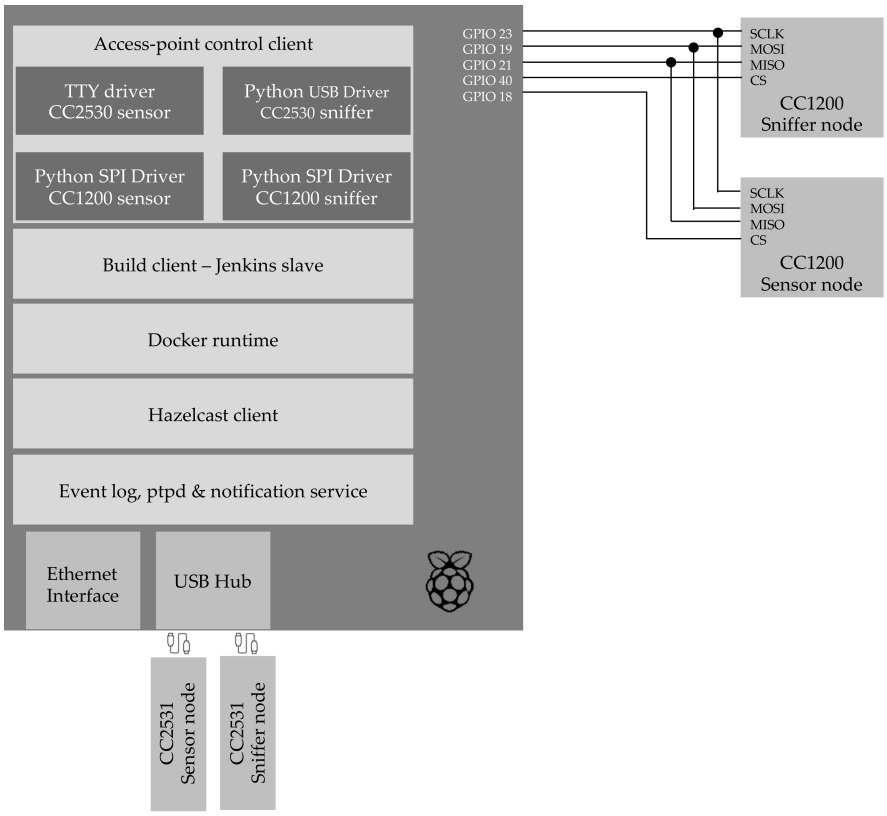

Fig. 3.11 Functional block diagram of access-point hardware

The RPi communicates in the sub-1 GHz band using the CC1200 radio front-end. There are two modules connected to the RPi over the GPIO headers requiring eight pins for each module. A four-wire SPI bus is connected with the three GPIO pins, each connected to a GPIO pin on the RPi. The CC1200 has a reset pin to reset using an input pin by toggling its state. This also resets the chip's register configuration, facilitating communication by a shared setting on the transmitter and receiver. There

are two modules with a SMA antenna. One of the modules is used for communicating with the field nodes. The other one is used for sniffing protocols missed by the access-point software due to the computing delay or state uncertainty on the CC1200 due to probable errors. The sniffer provides better insight by collecting messages in the air with timestamps to study the communication stability and reliability by the access-point software. There is a requirement to have well-coordinated timestamps in distributed systems i.e., all the collocated access-points deployed in the materials handling facility. Fig. 3.11 summarizes the hardware architecture with its software components of the PhyNetLab access-point.

3.3.3 Servers in PhyNetLab

The servers run the applications and web services, which consume data generated by the nodes in the PhyNetLab. They are placed in the topmost layer of the PhyNetLab architecture. In the implementation, this layer is located in a remote facility with enterprise-grade servers. These machines are capable of implementing high-level functions of the IoT application. They are connected over an IP network. In this implementation, servers are connected over the same subnet for reducing routing latency in the data-flow. Servers used in this layer either supplement in the operation of PhyNetLab or provide additional features for making experimentation possible. It also offers remote access for users to experiment on the industrial IoT platform with less or no human intervention. This layer on the PhyNetLab architecture categorizes three servers that provide storage, processing, and presenting the data generated through experiments or industrial applications in a real case scenario. Fig. 3.12 illustrates the configuration of the servers. It is not categorized if the server implementations are operated in containerized, virtualized, or dedicated servers. In the top layer of the PhyNetLab architecture, depending on the application and their availability, the servers are dedicated hardware for each application or sometimes virtualized. For example, a database server performs better on dedicated hardware utilizing the disk I/O performance and the memory exclusively for the database. In contrast, a web application performs equally on both dedicated and virtualized platforms. These servers' data traffic provides insight for the implementation as dedicated, virtual or container-based servers and applications.

Three physical servers are deployed to provide the necessary digital infrastructure for implementing the testbed. In total, there are 12 cores and 38 gigabytes of RAM available for the application servers. One of the servers has a Solid-State Drives SSD drive to improve the read/write access

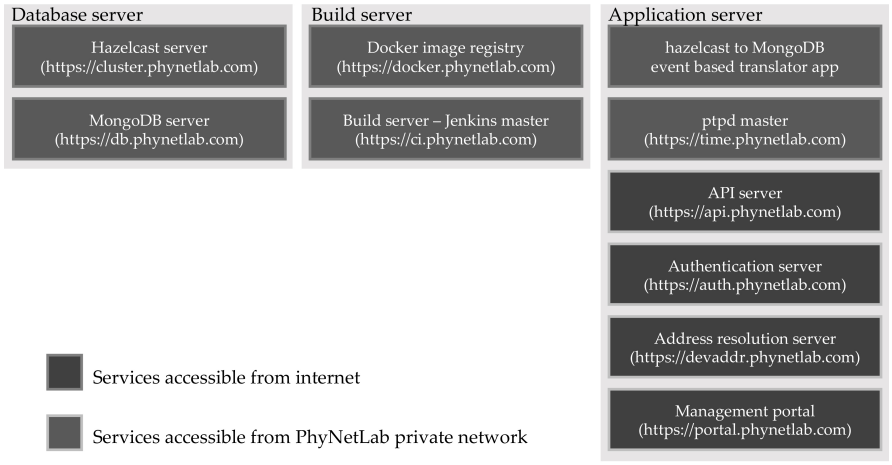

Fig. 3.12 Server categories of PhyNetLab

speeds used for the database server. Another server with a 500 gigabytes hard disk drive (HDD) is used for running application servers and automation servers. A server with 16 gigabytes of random access memory (RAM) is preferred in this case to provide enough space for running a large data store that is highly available and queried in real-time. A Synology DSM 1515+ is deployed with two active network interfaces, with up to 90 terabytes of disk space with 6 gigabytes of DDR3 RAM. These are the hardware components that provide the digital infrastructure for the different servers used in implementing the PhyNetLab testbed.

3.4 The Sensor Floor

In this section, an experiment infrastructure developed as part of this research is presented in detail. The main reason for developing another testbed is the time it takes to flash the energy-neutral nodes. The energy-neutral nodes are developed for application-based research with a focus on energy consumption. Therefore, another testbed was required for the rapid prototyping of physical layer characteristics on a large-scale. The Sensor Floor was conceptualized as a testbed and developed as an iteratively programmable testbed for dual-band application-based research. Architecture, software and hardware components of the experiment infrastructure are discussed in sections 3.4.1, 3.4.2, 3.4.3 respectively. The experiment infrastructure is a WSN testbed developed for evaluating industrial wireless sensing scenarios. It is an application-based testbed

where the architecture design supports rapid prototyping of wireless network scenarios with a high node density. In Sec. 3.4.3, a single node hardware overview is discussed in detail, followed by the Sensor Floor architecture. The architecture provides insight into the ability for data collection and rapid firmware prototyping for wireless network scenarios. The various software tools developed for the management of the testbed are detailed in Sec. 3.4.2. A 3D render of the Sensor Floor is presented in Fig. 3.13. The Sensor Floor is called so because 345 nodes are embedded under the research facility's wooden floor.

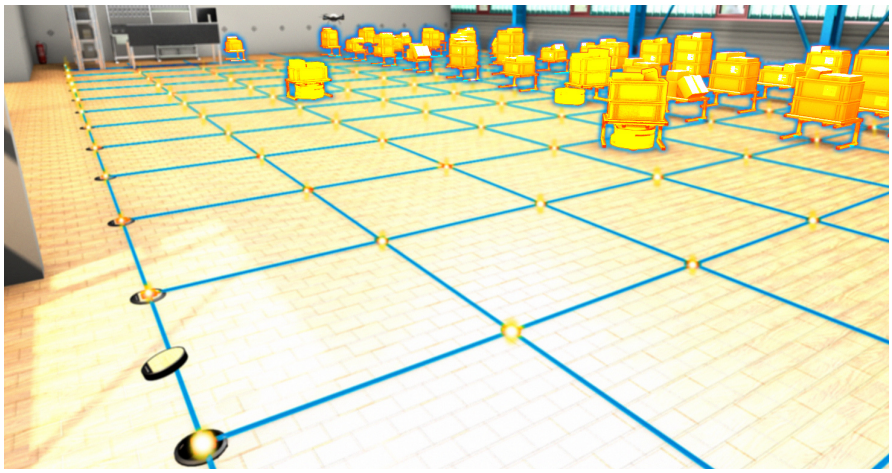

Fig. 3.13 Sensor Floor 3d render showing the topology of the nodes.

There have been multiple iterations of the Sensor Floor for various past applications, even though they were not called Sensor Floor. The applications of Sensor Floor had an array of floor mounted sensors that were capable of sensing physical changes on the floor. Remarkably, the two implementations by Prof. Dr. Joseph A. Paradiso from Responsive Environments, Media Lab, MIT, were the first few projects that drove the vision of Sensor Floor for interactive and sensing applications. The limitations in measuring physical phenomena were limited due to the technological limitations during the time of conception of this vision. The two following examples of Sensor Floor are examples of floor augmented sensor applications that used pressure-sensitive areas for measuring the interactions on the floor. An interactive floorspace has been developed in the first example, which uses modular uniquely addressable nodes connected to a sensor array of pressure-sensitive areas of varying size and shape [47]. It gave the potential to be integrated into an interactive environment [47]. The Sensor Floor was called the floorspace, which used

an array of force-sensitive resistors on each node to detect pressure, and the pressure information was output by way of a self-organized network formed by the floor nodes [47]. The pressure sensing and network systems were explored in this implementation of the Sensor Floor. It suggests potential applications of the floorspace in virtual reality gaming and artistic musical and dance performances in 2004 [47]. The other example is the Magic Carpet, which is a fusion of different sensing elements to capture human movement on a floorspace [48, 49]. It uses woven piezo-electric wires that were used for measuring change pressure due to footprint [48, 49]. A precursor to interactive floorspace [47] was the magic carpet, which improved the shortcomings of the Magic Carpet, such as individually addressable nodes that can be deployed in any shape or size. In the Sensor Floor proposed and developed here, the nodes are distributed, low-power wireless nodes with ten onboard sensors for measuring various physical parameters such as light, sound, mechanical vibration, and magnetic field fluctuations.

3.4.1 Architecture of Sensor Floor

The Sensor Floor has three components: the CC1350 STK, a carrier board where the communication bus and power lines are connected, and finally, a sink computer for data acquisition and firmware updates. The Sensor Floor is deployed in a 30-meter long hall with only 23 meter long useable areas. On the other side, it is 15 meters long, where we chose to deploy a cable of 15 meters with every meter for a sensor node. Every 15 meter length is called a strip composing 15 sensor nodes connected with power, 1-wire and RS422 communication lines. One end of the 15-meter line terminates with two USB dongles for 1-wire bus and the RS422 bus. The power supply is 12 volts powering each strip. In the end, there were 345 nodes in 23 strips, and 23 internet connected RPi. The dongles provided the necessary control for each node. The nodes can be used for data acquisition from the floor even though they have dual-band wireless communication interfaces to perform experiments. Each node can be flashed individually, and the whole floor can be flashed in parallel using a command-line tool developed exclusively for this deployment. The RPi sink computers are time-synchronized with a PoE connection. The data acquisition is tested to be synchronized within a few milliseconds within the allowable tolerance for data acquisition. The nodes' synchronization messages are delivered every 4 seconds on an average due to the design choices of using a 1-wire bus to apply changes to the communication bus.

3.4.2 Software components

There are two types of software for the stack to develop applications. In general, there are management tools that are developed, which also contain a boilerplate code for future experimentation. The second type is the application-specific firmware developed for the CC1350 hardware, with guidelines for flashing the nodes. The developed software is made available publicly at `https://github.com/akrv/sensorfloor`. The management software includes open source tools like the SBL flasher, which abstracts flashing for a CC1350 node using the serial boot loader interface. To enable the serial boot loader on the hardware, the 1-wire bus allows the back door pin for the MCU to understand that the power reset should boot into the boot loader mode. The boot loader mode is a standard specification from the hardware manufacturer (Texas Instruments), which allows the firmware to be flashed new depending on the experiment. The software is distributed so that all of the sink computers can run the same software, and their IP addresses are used to run commands remotely. A frequently run command is to synchronize the time between the sink computers before starting an experiment to reduce the skew in synchronization. This is called the *flash_floor* tool in the code repository. A GUI is developed to update the received data from each node of the strip, which is implemented using the flask library to allow for a straightforward REST API interface. This software also allows us to flash each node as peruse choice and even develop a custom flashing tool. The 1-wire bus status is exposed using the *owfs* software package, which abstracts the 1-wire communication into a set of file system changes. The *imu_reader* is a tool that contains a hex file targeted for the CC1350 platform. It reads the IMU data and the RSSI value of any received messages and transmits when an interrupt is received. This software also implements the boilerplate code for developing full-stack Sensor Floor applications. The 1-wire nodes are queried in a loop every 4 seconds, accounting for the 1-wire communication delays. The 1-wire communication involves turning off the previous node that communicated using the RS422 bus and turning on the next device in the strip to communicate. A hardware interrupt triggers the node to print into the serial interface, the data packets with information such as the IMU data and the RSSI data. Once the sink computer receives the messages, it is directly posted into multiple MQTT topics built with various unique identifiers, as shown in 3.2.

Listing 3.2 MQTT topic for Sensor Floor data acquisition

```
/imu_reader/<MAC address of sink>/<Node ID>
/imu_reader/<strip id>/<Node ID>
```

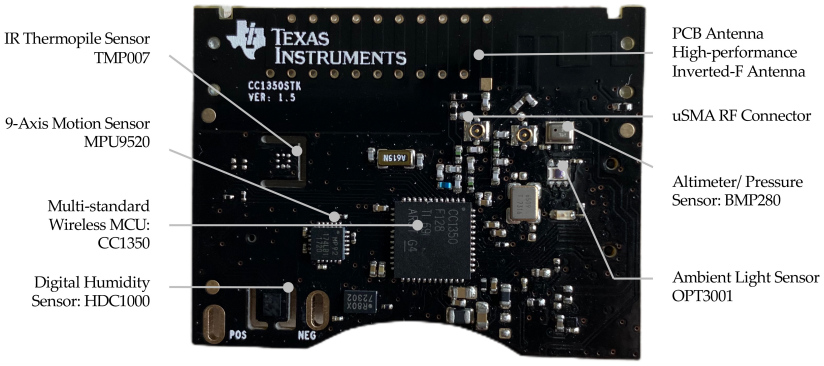

Fig. 3.14 CC1350 STK Version 1.5. Front side [51, 52].

This allows for platform-independent, language-agnostic distributed software architecture for developing Sensor Floor applications. MQTT is chosen since the library support for multiple applications is of many folds and can be easily integrated with any application. Another tool developed is the real-time REST API called the *summerschool_api* that uses a Redis in-memory database to persist strip data. A simple get request is responded to with a JSON formatted string that hosts all the payload information of each node and each strip and is available over the internet on-demand.

3.4.3 Hardware Overview

In logistics, there is a rising trend in adapting CPS and IoT devices to digitize industrial processes [9, 50]. A single node is chosen with fundamental hardware that can be used for future ultra low-power WSN applications. A single node is comprised of two boards, a sensor tag, and a carrier board. The sensor tag is an off-the-shelf product, and the carrier board is used for deployment purposes. This board ensures that the sensor tag can be supplied with power, data, and flashing capabilities.

3.4.3.1 Node

A Texas Instruments dual-band SoC CC1350 is chosen as the MCU for the single nodes. A sensor tag is a commercially available off-the-shelf IoT hardware that was developed for prototyping purposes. A sensor tag has up to 10 sensors (Ambient Light, Infrared Temperature, Ambient Temperature, Accelerometer, Gyroscope, Magnetometer, Pressure,

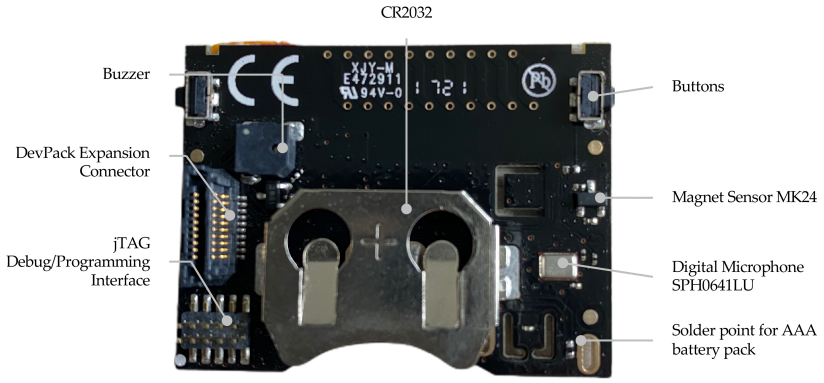

Fig. 3.15 Texas Instruments CC1350 sensor tag back side[51, 52]

Humidity, Microphone, Magnetic Sensor) that can be used for various sensing applications. Most on-board sensors are commercial standard for retail products such as the IMU. The hardware has a 15 pin mezzanine connector that is placed above the battery clip as seen in Fig. 3.15. The connector has all the required pins to power, communicate, control and flash new firmware on the host MCU CC1350. A debug board is also available for software development with the sensor tag. The debug board allows for run time debug using the code composer studio. This connector is used for connecting the carrier board. There are two hardware limitations for which workarounds were deployed. To flash the hardware using a serial bootloader (SBL), it was required to access special pins that were already dedicated to the on-board MEMS microphone. For accessing the SBL, special TX and RX pins have to be used rather than typical UART TX and RX pins[53]. DIO 12 and 13 are used UART_RX and UART_TX respectively [53]. Since DIO 12 is used for AUDIO_DI, as illustrated in the hardware schematic Fig. 3.16, the solder pads of the 0-ohm resistor are used to jump a cable for UART pins of the SBL. The configuration of the SBL also requires another trigger to boot the MCU into bootloader mode. This happens by checking if a specific DIO pin is in a configured state (active low or high) [53]. Therefore, a logic level switch is required to differentiate every boot of the MCU between a flash or a normal reset. These are the two limitations that are overcome using a carrier board that would only be used for delivering power for the sensor tag.

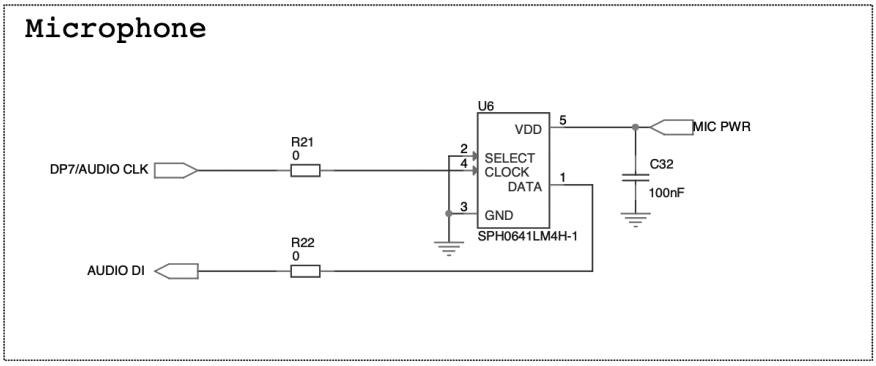

Fig. 3.16 CC1350 sensor tag MEMS microphone tied to SBL pin [51, 52]

3.4.3.2 Carrier board

The carrier board is required to power up, communicate, and finally flash the sensor tag for iterative experimentation due to the limitations mentioned earlier. The carrier board also has to perform logical operations to set a pin high or low depending on the operations' order. If the sensor tag will be flashed, it is required to set the BL_PIN_NUMBER to the pre-configured active state. Since there are 15 carrier boards connected in series per strip, a 12 v supply is used. However, the source voltage for the sensor tag is 3.3 v. The power circuit is designed with a step-down converter, as shown in the schematics in Fig. 3.17. The 5 v source is used for the 1-wire interface and the multiplexers that perform, switching the power off and switching between the UART pins for flashing and regular communication. A low-dropout regulator is used as a voltage source for the 3.3 v of the CC1350 sensor tag.

To overcome the limitations of the SBL due to the special pins used for flashing using UART, two different pairs of communication channels are used for flashing and for regular communication. A communication multiplexer (TVHD1552) is used to switch between the two lines of communication. A 1-wire interface (DS2408) has 8 open-drain digital pins used to switch the multiplexer, communication interface converter, and power source. Since there are 15 carrier boards per line, each 1 meter apart, UART is not functionally usable. Therefore the carrier boards convert to a full-duplex differential serial communication interface RS-422. The RS-422 interface on the carrier board is also controlled using the 1-wire interface to turn on and off the communication. This enables a token-ring based communication topology between the 15 devices in a single strip where the token is controlled using a sink device at the end of the strip. Here, the sink

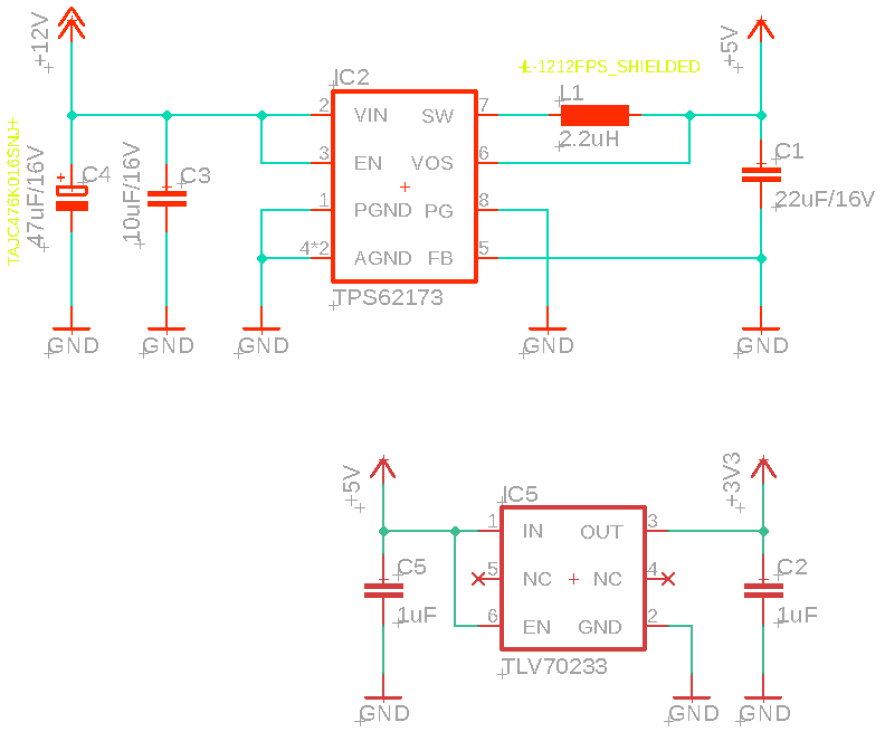

Fig. 3.17 Carrier board power distribution for the CC1350 sensor tag pin

device is RPi, which connects to the internet and relays the RS-422 interface's information.

3.5 Decentralized Brains: Concept and Design

In this section, the characteristic of *Decentralized Brains* is explored, and the protocol is designed and developed. After developing the fundamentals of the *Decentralized Brains*, they are evaluated using the Sensor Floor. Contributions to this concept are (*i*) a networking architecture using well-proven, low-power, low data-rate industrial CPS and wireless communication standards applicable for self-assembly protocols, (*ii*) a novel method for creating *Decentralized Brains* at every node through data replication for self-assembling swarms and (*iii*) a proof-of-concept implementation on commercially available off-the-shelf hardware along with a large-scale performance evaluation is presented to prove the viability

of developed communication primitives for scenarios in Industry 4.0. Finally, in Sec. 5.1, a discussion of space applications with the help of TESSERAE provides a detailed insight into the various scenarios and features of *Decentralized Brains* that can be used in swarm behavior and space applications.

The distributed cognition framework in [54], for collaborative efforts between humans, defines distributed cognition as cognitive activities viewed as computations that occur via the propagation of various information and knowledge transformation through media. The media here refers to internal (e.g., individual memories) and external representations (wireless communication and other sensing capabilities). The states of the representations refer to various information and knowledge resources transformation during collaborative maneuvers. According to the study [54], the way knowledge is propagated across different representational states. It is characterized by communicative pathways that are continuously interrupted and coordinated sequences of action by demanding an ever-changing environment. Here, the evolution of the nervous system in an octopus is used as an inspiration, and the notion of distributed cognition is used to conceptualize the idea of Decentralized Brains. The representational states are reliably replicated between the collaborating systems in a distributed wireless communication architecture [55]. This allows for local actuation without global coordination and facilitates multiple parallel control for collaborative maneuvers [55].

In distributed systems, the most common replication topology is to have a single leader that then replicates the changes to all the other nodes in the network with the benefit of avoiding conflicts caused by concurrent writes [55]. All the clients are writing to the same server, so the coherence in the data is maintained by the leader [55]. As shown in Fig. 3.20, our data replication spans two physical layers in radio communication with multiple state transitions starting from network discovery. This helps in reducing channel contention while keeping lower bounds in latency [55].

The consensus protocol is inherently developed as a centralized protocol with network discovery and leader election methods. This architecture's primary motivation is to understand and identify a consensus protocol's workhorse and develop those networking primitives in a modular manner reliably. Modularity and reproducibility of the results in WSN will facilitate in developing different complex consensus algorithms. From understanding and implementing a decentralized context broker [56], it can be drawn out that the fundamental requirement for a consensus protocol to reliably function is to have a robust atomic broadcast communication primitive within the network. Therefore, a method of reliable network flooding is identified in [55] and is implemented considering the target hardware. An

experiment is setup to prove that the broadcast communication primitive can be reliably deployed in Sec. 3.5.8.

The communication flow of the *Decentralized Brains* is illustrated in Fig. 3.20. There are two types of nodes in the network, the leader node, and the replica node. Leveraging that each node is equipped with a dual-band networking, it is possible to operate two networking paradigms in parallel. One of the networks is deployed in 2.4 GHz networking, a 6LoWPAN network. The other network is deployed in the Sub-1-GHz network, which performs the data replication operation among the nodes [55]. A cohesive network spanning over two radio networks is developed in *Decentralized Brains* to increase the communication throughput and decrease the latency for replication [55]. The data replication state is met when networking flooding is performed using the synchronous broadcasts to propagate the same message across many nodes that are sparsely deployed in a large spatial area or in a dense environment [55].

In Industry 4.0 application scenarios, it is considered that the IoT devices and other machines are located in a dense environment, i.e., the number of devices per square meter is considerably high compared to smart city IoT applications. The data that needs to be replicated is sent from the originator of the data to the leader node [55]. The leader node initiates the synchronous broadcast, which results in a network flooding in the sub-1-GHz band [55]. The leader node also receives the replica changes, and it manages the most recent replica of the data structure that will be propagated throughout the network using the synchronous broadcasts [55]. For synchronous broadcasts, strict clock synchronization is required between the nodes, which is implemented and tested in Sec. 3.5.8. 6LoWPAN is the networking layer provided by Contiki-OS, which is used to create the 2.4 GHz network for one-to-one communication between the originator data and the leader node [57]. This networking layer provides a multi-hop mesh network. Therefore the originator and leader nodes can reliably communicate in harsh environments [57]. Since there is an extensive amount of literature available discussing the reliability and performance of a large-scale 6LoWPAN based networking, this work aims to present the synchronous broadcasts' performance evaluation, a newly developed networking primitive for this hardware.

3.5.1 Background

When communication between multi-part systems is desired and needs to only support low-rate data transfer, Wi-Fi is usually eschewed for consuming too much power in favor of mesh-networking standards like

ZigBee [58] or BLE [59]. The trade-off that needs to be met with these existing wireless communication standards is between energy and scalability. Wi-Fi and ZigBee work in the frequency range of 2.4 GHz with 11 MBit/s and 250 KBit/s data-rate. Additionally, the number of nodes in an area is fairly limited to tens of Wi-Fi devices, to a few hundred machines in ZigBee. The amount of energy used for operation, the density of nodes that can work in an area, and the network topology of these standards have various features that can be used according to the application and the system requirements. We will use the umbrella term of WSN and the IEEE standard 802.15.4, representing the Low-Rate Wireless Personal Area Networks (LR-WPAN). For example, ZigBee is one of the communication protocols in this standard. We propose using this standard due to energy and data-rate requirements and use self-healing and scalability cases that are well matured in this standard.

Wireless is a broadcast medium (i.e., when a node is transmitting, all the other nodes can listen, but when another node also transmits, both the transmissions are lost due to packet collision). There are recovery methods for MAC by using a random backoff delay. Additionally, a MAC protocol called CSMA / Collision Avoidance (CA) is used in a time-slotted manner in IEEE 802.15.4, where nodes sense the transmission medium duration for any transmissions and reschedule if the medium is busy [11, 58]. Since a delay is applied to every transmission at MAC layer, scaling the number of nodes communicating in the medium increases packet delivery latency [11]. When swarm behaviors are implemented, certain states and data must be communicated globally between the nodes in a *guaranteed time*. Instead of waiting for the medium to be free, we leverage that every node in the network can hear the transmission. Consequently, we flood the medium constructively with data that needs to be propagated. The same packet's simultaneous transmissions interfere constructively with respect to the specific physical-layer characteristics, allowing receivers to decode the packet with high redundancy and achieving high-flooding reliability that approaches the theoretical lower latency bound across diverse node densities and network sizes [60]. The phenomenon of constructive interference to achieve the synchronous broadcast effect is precisely explained in Sec. 3.5.1.1 and synchronous broadcasts are described in Sec 3.5.3.

Using high-accuracy message timestamping and a combination of low and high-frequency clocks, VHT synchronization between the nodes can be achieved [61]. Since time synchronization occurs by message timestamping and the messages transmitted are always of the same length, clocks among the nodes can be kept synchronized across multiple hops irrespective of the network's number of nodes. This kind of clock synchronization allows temporal decoupling of synchronous transmission, which frees the medium

and the MCU between each communication round. Therefore, a modular communication stack and increasing the throughput of data replication among the nodes are feasible.

Contiki-OS provides a modular application program interface where multiple communication stacks can be run using the same RF core [57]. With new progress and increasing market demands for WSN in IoT applications, there are various SoC products available. The two kinds of SoC that we used for the proof-of-concept implementation are MSP430-CCRF [62], which uses an MSP430 MCU core with a CC1100 RF front-end working in the sub-1GHz band and CC1350 which is a dual-band SoC with an ARM M3 architecture MCU. The latter mentioned MCU has 2.4 GHz operation as well as a sub-1 GHz frequency band for communication. This helps in splitting the traffic between two physical layers simultaneously while keeping the energy requirement low.

3.5.1.1 What is Constructive Interference?

Constructive interference is a phenomenon that is extensively used in the development of *Decentralized Brains*. This was used to reliably flood the network with the same payload across all the nodes. A physicist might innately understand constructive interference as an effect where the electromagnetic waves are phase synchronized to have positive proportional effects. However, propagation's positive effect is indeed used; it is not used in the same way as it might be understood. Constructive interference, as described by Marco Zimmerling et al. in the 2020 survey on synchronous transmissions in low-power wireless communication, is the following: When all signals arrive with similar power as transmitted, a successful reception is only possible if the transmitted packets are identical [63]. This is when the effect of constructive interference occurs [60] [63]. As mentioned earlier, real constructive interference can be understood as a pure overlap in the signals where there are no time and phase offsets of the received signal [63]. However, even if transmitters would compensate for different path delays during transmission, in an effort for the signals to arrive without the time and phase offsets at the receiver, the phase offsets would vary throughout the reception because of different carrier frequencies across the transmitters [63].

In electromagnetic radiation-based packet propagation, especially in IEEE 802.15.4 radio networks, while communication is taking place in a channel other devices are forbidden from communicating during the same time to avoid packet collisions. In case a node radiates its message while another node is also radiating in the same channel, packet collision is imminent due to the interference of the carrier waves. There are extensive studies on the

effects of such interference and various scheduling mechanisms to mitigate radio-based interference issues. This effect of interference is destructive as packet collision results in loss of transmission of both the packets. Therefore, simultaneous radio propagation results in destructive interference have been state-of-the-art for a long time until Glossy [60] was proposed for network-wide packet flooding using constructive interference by Macro Zimmerling et al. [60, 63].

Constructive and destructive interference: Interference is constructive when the receiver senses the superposition of the base-band signals produced by multiple transmitters [60]. Interference is normally termed as a destructive effect, as it prevents the receiver from detecting and demodulating the superimposed base-band signals correctly [60]. Constructive interference has not been widely exploited in low-power, low data-rate sensor networks due to the involved complexities and limitations due to available energy or hardware capabilities. Two of the limitations are that it requires sufficiently accurate clock synchronization between all the nodes in the network and the requirement for a highly predictable software delay [64] [60]. Instead, some protocols exploit the capture effect [65] that occurs when a wireless radio senses a signal from one transmitter despite interference from other transmitters [60]. The capture effect, also known as *co-channel interference tolerance*, is the ability of certain radios to receive a signal from one transmitter despite interference from another transmitter, even though the relative power of the two signals is almost the same [66]. A radio can detect one signal when it is stronger than the others (power capture [66]) or when it begins to be received slightly earlier than the others (delay capture [67]) [60]. However, the capture effect suffers from scalability problems when multiple transmissions overlap, leading to packet loss as is always feared in the case of interference [60, 65]. Requirements for constructive interference are highly dependent on the communication scheme, in particular on modulation and bit rate [60]. We derive the upper bound for temporal displacement between several concurrent transmissions of the same packet, which allows the packet to be correctly received with high probability due to constructive interference [60] which is performed as an experiment. The results are presented in Sec. 3.5.8.

The effect is achieved due to strict time synchronization, leveraging the type of carrier wave modulation used in the communication standard IEEE 802.15.4. If the same payload is radiated in the same channel with an allowable time delay, the receiver can decode the message even though the packets interfere with each other. For example, the standard IEEE 802.15.4 transceivers using the DSSS can tolerate twelve incorrect bits (commonly known as chips) per symbol. This is an error-correction capability to receive packets allowing a lower threshold for interference and offsets in the carrier frequency. With the help of literature [60, 68] and by conducting

experiments in target hardware as presented in Sec. 3.5.8, the claim that an allowable temporal displacement of 0.5 μs between the transmitting signals allows for a successful reception is validated and further used in the protocol design of *Decentralized Brains*. This is the effect of constructive interference that is used for network-wide packet flooding. The packet flooding using constructive interference is called *Synchronous Broadcasts*, where the same message is broadcasted by multiple nodes at the same time interfering constructively in the channel allowing the receiver to decode the message. The specifics of constructive interference and the protocol of Synchronous Broadcasts is detailed in Sec. 3.5.3 after the different elements that are used for achieving synchronous broadcasts such as state synchronization, data replication, Contiki-NG, and time synchronization are introduced in Sec. 3.5.1.2, 3.5.1.3, 3.5.7, 3.5.2, respectively.

3.5.1.2 State Synchronization

In ad-hoc mobile deployments in extreme environments, it is often necessary to update all the nodes in the network, such as re-joining the network or changing system parameters as requirements and network conditions change.

Inconsistencies between the nodes in terms of data and the physical state could lead to undesirable events, which could be detrimental to the network; therefore, reliable data dissemination is required. There are protocols such as Deluge and Splash [69], which wait for an explicit request when a node misses an update and [70] proposes a better approach by using synchronous transmission of the nodes and setting the flag in the payload as an implicit confirmation [55].

Global configuration management is a feature where all nodes in a network can be configured to operate coherently. In specific applications, the configuration is a piece of global information about the system's state, such as changing the nodes' duty cycle to wake up at a different schedule or delay a particular function according to its physical position but propagating the same delay configuration throughout the system. Here the decision to act is made locally in the nodes depending on the physical parameters. The reliable information dissemination has a contingency that when a node does not receive the information during the broadcast for replication, it can poll the leader to receive the information [55].

Aggregate functions in database management is where the values of multiple rows in a relational database or a value for a key in all of the documents in non-relational databases are aggregated to a single value. Aggregation is performed depending on an operator applied to them, such as sum, average, or median. Aggregation is considered the most common

operation in sensor networks. They are computing the maximum (or minimum) where nodes merge the flags by taking the bit-wise OR and the payload by taking the larger (or smaller) number. Developing operators such as duplicate-sensitive aggregates are complex. When a packet is dropped, or a packet is received more than once, an error in the partial aggregate occurs. A reliable final aggregate can be delivered to all participating nodes in the network with presented data replication schemes [55].

In **network-wide agreement**, the nodes have to agree on certain parts of information to perform their actions. A network-wide agreement might be required in performing tasks for swarm behavior. When the data is replicated in all the nodes, the nodes can perform coordinated tasks like the example drawn from an octopus distributed brain and its evolution. **Atomic broadcast** is considered as a vigorous embodiment of network-wide agreement. In this approach of *Decentralized Brains*, log replication is proposed instead of leaderless replication techniques such as 3-PC and 2-PC, as proposed in [70]. All the network nodes receive a time-slotted broadcast message with more reliability since every change is not propagated. The leader of the network initiates this message. Any change in the node is communicated to the leader in the network. During the following broadcast round, the data is propagated to the rest of the network nodes [55].

Modular communication patterns The communication architecture supports multiple communication patterns, including all-to-one and all-to-all traffic, where nodes would merge by inserting their data into reserved parts of the payload field. Due to the limited packet size in IEEE 802.15.4, multiple communication patterns can be used within the time-slotted data replication scheme proposed. Moreover, a mesh network spanning a diversified physical layer makes the system reliable and provides modular communication techniques that could be used at different stages of an application. Added to this architectural advantage, the Contiki-OS offers a more comprehensive and flexible programming interface for wireless nodes [57], which is chosen as the target application framework for implementing proposed data replication schemes with diversified physical layer radio. In Sec. 3.5.7, the basics of Contiki-OS and the reasons for identifying it as the target application framework are presented [55].

3.5.1.3 Data Replication

In *distributed computing*, state machine replication or data structure replication are standard methods for implementing a fault-tolerant service by replicating servers and coordinating client interactions with server

replicas. In micro-services architecture for a web-based application, data replication schemes have been studied vastly, starting with the conception of a protocol called Paxos, which was developed by Leslie Lamport [71]. Raft is a consensus algorithm for managing a replicated log. It produces a result equivalent to (multi-)Paxos, and it is as efficient as Paxos, but its structure is different from Paxos; this makes Raft more understandable than Paxos and also provides a better foundation for building practical systems [7]. Therefore, much of the work was inspired by Raft for the conception of a data structure replication scheme in WSN [55].

The three main pillars for any distributed data system are consistency, availability, and partition tolerance. Consistency means that all the nodes in a cluster see the same data at any given time. Availability means that reads and writes will always succeed, even if we cannot guarantee that it will have the most recent data. In practice, it means that we will still be able to use one of our databases, even when it cannot talk to the others. Partition Tolerance means that your system will continue working even if there is a network partition. A network partition means that the nodes in your cluster cannot talk to each other. The CAP theorem stands for Consistency, Availability, and Partition Tolerance, which states that given these three properties in a distributed system, an algorithm needs to prioritize any two of those because a system cannot have all three properties. The most common replication topology is to have a single leader that replicates the changes to all the other nodes in the network to avoid conflicts caused by concurrent writes. All the clients are writing to the same server, so the leader node maintains the data's coherence. In low data-rate WSN systems, the delay between the time an update is applied in the leader node and the time it's applied in a given replica is unavoidable due to the communication channel's limitations. This delay is called the replication lag, which causes inconsistencies in the data replication scheme. This can be mitigated by several means to achieve eventual consistency. Additional to replication, there is consensus in distributed systems that reach agreement among several processes about a proposal, i.e., accept or decline it after a finite time of execution. Achieving consensus becomes challenging when faults may occur when communication is lossy, and nodes may crash [55].

Two widely used yet simple consensus protocols are 2-PC and 3-PC. We introduce both the mechanisms for consensus and discuss their respective properties and limitations, as presented in [70]. The protocol assumes the existence of one static coordinator and a set of participants, or cohort. As the name suggests, 2-PC works in two phases: (a) Proposal Voting: the coordinator broadcasts a proposal to the cohort, each member replies with its vote, yes or no; (b) Decide: the coordinator decides to commit if the vote is yes unanimously; otherwise, it decides to abort. It then broadcasts the decision to the cohort that will commit or abort upon receiving the message.

2PC is simple and comes at a low communication complexity, but has a significant limitation of being a blocking protocol. Whenever a node fails, other nodes will be waiting for its next message or acknowledgment indefinitely, i.e., the protocol may not terminate. Recovery schemes can be considered but fall short when it comes to handling two or more failing nodes. In particular, if the coordinator and a participant both fail during the second phase, other nodes might still be in an uncertain state, i.e., have voted yes but not heard the coordinator's decision. If all remaining nodes are uncertain, they cannot make a safe decision as they do not know whether the failed nodes had committed or aborted [70] [55].

3-PC mitigates the above limitations by decoupling decision from a commit. This is done with an additional pre-commit phase between the two phases of 2PC. The three phases are as follows: (a) Proposal Voting: same as in 2PC; (b) Pre-Commit (or abort): the coordinator and participants decide as in 2PC, but no commit is applied (abort is applied immediately); (c) Do Commit: participants finally commit. The additional phase guarantees that if any node is uncertain, then no node has proceeded to commit. The protocol is non-blocking in a single participant node failing: remaining nodes time out and recover independently (commit or abort). 3-PC can also handle the failure of the coordinator and multiple nodes by using a recovery scheme. Nodes will then enter the termination protocol, communicate and unanimously agree to commit, abort, or take over the coordination role and resume operation. In the more challenging case of a network partition, 3-PC is unable to maintain consistency. The design relies on synchronous transmissions for efficient communication and on merge operators for enabling various all-to-all interactions. In [70], the principle of synchronous transmission of broadcast messages for flooding in the medium, as presented by [60] is exploited. Since this reliable broadcast scheme is beneficial for replication, we also leverage this scheme in replicating our data structures [55].

3.5.2 Time Synchronization

When a node broadcasts a message to two nodes, the node receives the messages in different timings according to their spatial displacement when considering a free space propagation model. The time between the nodes can be estimated as close as possible to the propagation delay, which is the only scope of offset in free space propagation. As shown in figure 3.18, the spatial displacement is apparent by the delay in the reception between the nodes $RX_2 - RX_1$. Since the time stamp occurs not at the rising edge as shown in the Fig. 3.18, there is a SFD used by most transceivers where an

interrupt occurs for a reception. This interrupt is used for timestamping. Even though this could be a source for errors, we eliminate this source to keep the length of the payload constant for the data replication primitive. When a time synchronization algorithm relies on timestamping the reception of a specific message at the SFD, it is inherently prone to an offset of 160 ns for 2.4 GHz band as presented in [61]. As mentioned in Sec. 3.5.1.1, the synchronization is necessary to minimally align the temporal distance for the receivers to decode the information while ignoring multiple instances of the closely aligned signal as reflected signals. For space applications, when the distances are large, as in the case of CubeSats, then errors must be mitigated by distance estimation. VHT synchronization for low-power, low data-rate wireless sensor networks makes them reliable for coordinated tasks [61] The two critical factors needed to achieve precision time synchronization are:

1. a high-resolution clock source
2. message timestamping with high accuracy

The two basic requirements for achieving low-power time synchronization are: [61]

1. low-frequency clocks
2. infrequent communication.

The requirements mentioned above do not comply with operation in low-power. The higher a source oscillates, the higher the leakage current, and high accuracy message timestamping is dependent on this clock source. The basic idea behind VHT is that during active periods, the high-frequency clock is turned on, and a hardware counter counts the number of high-frequency clock ticks. There are high-frequency clock ticks during each low-frequency clock tick [61]. When an event of interest occurs, like the SFD

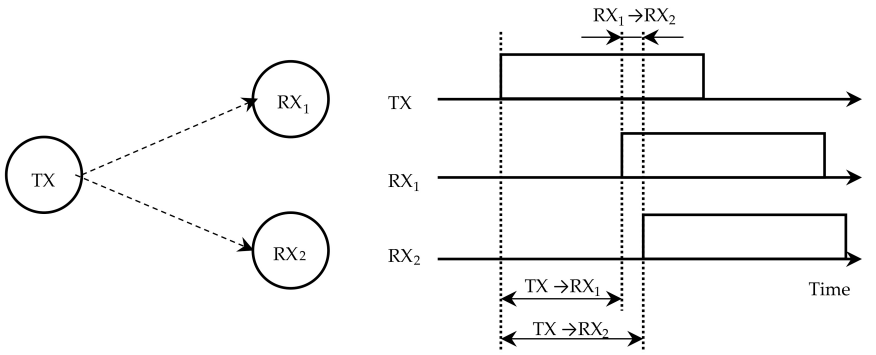

Fig. 3.18 Scope of offset in time synchronization [61]

event, the system records not only the value of the counter sourced by the low-frequency clock, but also the value of the counter sourced by the high-frequency clock. The high-frequency clock counter is reset at the end of the most recent clock ticks of the low-frequency clock to keep up with the limitations of counting in a 32-bit MCU architecture [61].

$$\varphi_0 = \frac{f_H}{f_L}$$

Thus, this second timer will indicate the *phase* φ within a low-frequency clock tick, allowing the effective resolution to be up-sampled to the high-frequency clock (modulo one cycle of jitter) [61]. The event time is sampled as

$$t_{event} = C_L \cdot \varphi_0 + \varphi.$$

Two capture units, one on each timer, are connected to the SFD interrupt line. An additional capture unit on the fast timer captures the counter on every low frequency rising edge. The value of h_0 is stored in one of the capture units. Later, when the SFD line rises, the two capture units on the two timers store l_0 (the value of the low-frequency counter) and h_1 (the value of the high-frequency counter), respectively. The event time can be calculated using these three captured values as

$$t_{event} = l_0 \cdot \varphi_0 + (h_1 - h_0) mod \varphi_0.$$

The modulo operator is necessary to compensate for the speed of the MCU and relaxes the timing constraint since high-frequency ticks are at an interval of φ_0 . While overflows of the high-frequency counter are not a problem, the low-frequency counter's overflows must be dealt with in software to provide virtualized system counters of higher widths. Also, while 32-bit counters may be adequate for low-frequency 32 kHz signals, such timers overflow every five minutes if used for VHT with a high-frequency clock of 8 MHz, suggesting that 64-bit counters will be needed to support long time intervals [61]. The VHT algorithm also suggests that high-frequency clocks are needed not just for time stamping the reception or transmission of a message with a high time resolution, but they are also required to improve the frequency error estimation over intervals or the greater the clock frequency, the shorter the interval of time needed to synchronize a pair of clocks to a given frequency error resolution [61]. This could be very useful when applications require greater accuracy. The system clocks can be increased during design time to synchronize at a given frequency error resolution [61].

3.5.3 Synchronous Broadcast

When a node is transmitting in a wireless medium, the node is broadcasting the same packet information in the allocated channel for all the nodes in the network. Due to this broadcast nature of wireless communications, interference occurs whenever stations are close enough to listen to each other and whenever they are transmitting concurrently [60]. Collision or interference is an effect due to the overlap of signals in both the time and space when the transmitting nodes share the same physical layer characteristics [60]. Digital wireless communication implements multiple factors for redundancy in radio communication to mitigate interference. Therefore, the communication link reliability becomes a probability in the success of transmission from a Boolean if the communication will be successful or not for a given scenario. Interference is one such factor that reduces the probability that a receiver will correctly detect the information embedded in the wireless signals [60]. Interference has always been considered destructive, owing to the fact that the communication reliability reduces due to interference. There are two types of effects due to interference which can be called *constructive* and *destructive* interference [60].

We explore the nature of constructive interference in this case to develop the most crucial communication primitive of *Decentralized Brains*. If the overlap of the transmitted signals does not superpose, then the interference is destructive, and the probability of the transmitted data reaching the destination reduces. Whereas, when the base-band signals from multiple transmitters superpose, the receiver detects the superposition of the transmitted signals generated by multiple transmitters [60]. For achieving the effect of constructive interference, the transmitted base-band signals must be within a time window with respect to the carrier frequency used for communication to allow for the detection of superposition. The time window is strict as the mismatch in temporal synchronization will affect the transmission reliability due to destructive interference. To achieve a constructive effect, the transmitting stations must be time-synchronized. Due to the complexity, the cost of energy, and the hardware support for reliable software execution were required for implementing time synchronization protocols, the effect of constructive interference has not been extensively exploited in WSN [60, 64]. In [60], a low-power time synchronization algorithm is implemented to achieve the effect of constructive interference, whereas in *Decentralized Brains*, a different approach leveraging the special hardware capabilities is used, which is discussed in Sec. 3.5.6. Using the time synchronization, we enable nodes to synchronize their clocks required for waking up and listening to a broadcast, calculating any skew, correcting the clock, and waiting for the predetermined delay to re-transmit. By synchronizing to another transmitter

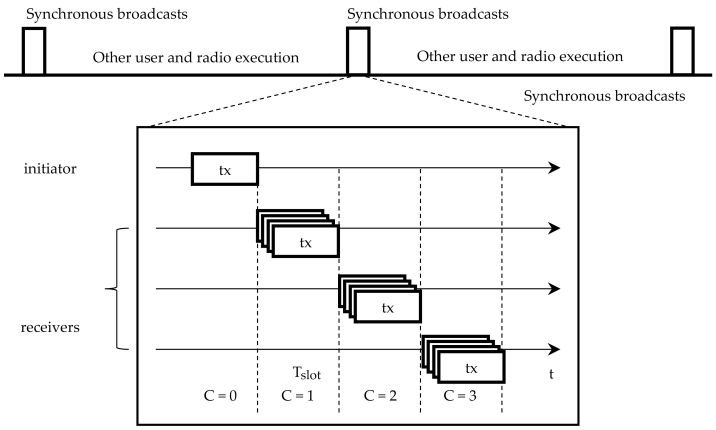

Fig. 3.19 Synchronous broadcast rounds with T_{slot} and counter C for number keeping track of number of transmissions [55, 60]

clock, we exploit the nature of constructive interference, allowing multiple concurrent transmissions to occur on the same channel. Using CSMA/CA MAC strategy, the broadcast medium is used efficiently for contention-based wireless transmission by avoiding any collisions of two transmitting stations in the medium. But to allow for multiple concurrent transmissions using constructive interference, we enable the devices to transmit into the wireless channel all at the same time. Even though it is contrary to the well-proven CSMA/CA MAC strategy, we make it possible for the devices to transmit simultaneously with the help of strict time synchronization. By creating the opportunity for nodes to overhear packets from neighboring nodes using [60], nodes turn on their radios, listen for the transmitted packets over the wireless medium, and relay overheard packets immediately after receiving them with an allowed software delay where time synchronization is performed amongst the nodes. Since the neighbors of a sender receive a packet at the same time, they also start to relay the packet at the same time. Here the time at which each node transmits after the reception is governed by the time synchronization. Nodes are allowed to transmit from the set of received messages that are defined in the communication protocol. This again triggers other nodes to receive and relay the packet. In this way, *Decentralized Brains* benefit from concurrent transmissions by quickly propagating a packet from a source node (initiator) to all other nodes (receivers) in the network [55, 60]. Based on [55, 60], the temporal offset among concurrent IEEE 802.15.4 transmitters must not exceed 0.5 µs to generate constructive interference with high probability.

As shown in Fig. 3.19, the radio transitions between three states are looping between two states until the counter is exhausted. When a new

node joins the network without using the network discovery, the node synchronizes its clock with the root node by listening to the broadcast. The broadcast frame carries a counter used to determine the number of transmissions the node has to make with the same packet. Once the counter is 0, the node goes to sleep until the next broadcast round.

3.5.4 Protocol definition

In *distributed computing*, state machine replication or data structure replication is a standard method for implementing a fault-tolerant service by replicating servers and coordinating client interactions with server replicas. The fundamental communication primitive used in distributed computing to achieve consensus between processes is the atomic broadcast. Atomic broadcast is where all correct nodes in a network receive the same set of messages in the same order. The broadcast is termed *atomic* because the same set of messages are received by all the nodes in the same order i.e., same sequence of messages are received by all the nodes [55]. To adhere to

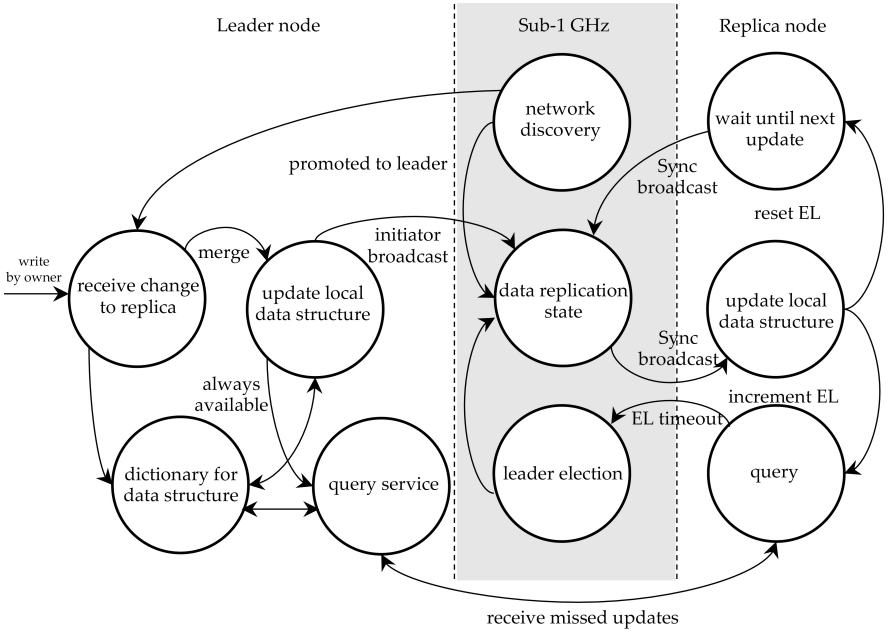

Fig. 3.20 Decentralized data replication networking architecture with diversified physical layers [55]

the nodes' energy requirements, provide a guarantee on data dissemination, and reduce the number of nodes competing to transmit in the medium, a time-slotted atomic broadcast is used. Using the time synchronization from the VHT algorithm [61], we enable nodes to synchronize their clocks as close as required to create constructive interference. Using the time synchronization, we enable nodes to synchronize their clocks to wake up and listen to a broadcast, calculate any skew, correct the clock, and wait for the predetermined delay to retransmit. Our atomic broadcast is designed using Glossy [60] to leverage synchronous transmissions, which have been shown to boost the performance of *network flooding*. Glossy provides reliability above 95% in a scenario where the capture effect does not occur; while varying the number of concurrent transmitters between 2 and 10, reliability stays relatively constant and always above 98%. It achieves an average time synchronization error of less than 0.4 μs even at receivers that are eight hops away from the initiator [60]. Instead of adding random back-off delay to the transmission in CSMA and creating uncertainty in the upper bound guarantee for swarm behaviors, the nodes transmit the same packet aggressively for a short period creating constructive interference and propagating the message network. When there are 94 nodes in a network with eight hops, the latency of propagation is less than 3 ms, including synchronization error and injected software delay between two transmissions [60]. The design of data replication for *Decentralized Brains* using a multi-hop mesh network with various physical layer relies on synchronous transmissions for atomic broadcasts [55]. The theory of constructive interference is detailed in Sec. 3.5.1.1 and the action of performing a synchronous broadcast is detailed in Sec. 3.5.3.

As shown in Fig. 3.20, our data replication spans two radio physical layers with multiple state transitions starting from network discovery. This helps in reducing channel contention while keeping lower bounds in latency. In network discovery state, the node first joins or creates a network depending on the functions of the node. Network discovery is used for leader discovery for a new node joining the network [55].

As shown in Fig. 3.22, the network discovery state terminates in a data replication state where the node promotes itself to a network leader or follows another leader. The time for leader discovery depends on the next broadcast slot, or it will timeout at the *discovery timeout*. If a node promotes itself to leader status, then the node is the initiator of synchronous broadcasts [55].

In an ideal state, reliable nodes with strong links are desired to be the leader. Leader election is triggered only as a fault-tolerant mechanism. When a leader in a data replication state misses more than a predetermined number of broadcast rounds, all the nodes become a candidate for election. The *election trigger* is the number of missed rounds, that when multiplied by

the duration between broadcast rounds, serves as the discovery timeout used in leader discovery [55].

As shown in Fig. 3.23, the leader election is similar to a bully election algorithm [72] except for one assumption that every node address is known by the other, which is mitigated by a user-set priority. The election state lasts for a duration from the last missed broadcast round to the upcoming round. This duration is split into consecutive synchronous broadcast rounds where the initial three rounds are reserved for the nodes with user-set priority. The rest of the slots is randomly used by nodes or depending on an objective function for selecting a back-off delay. The first node sends a *leader announce* message, becomes the leader, and the participating nodes re-transmit this broadcast to establish the leader. The leader transition to a data replication state happens after propagating the first broadcast message as the new leader [55].

In the data replication state, the nodes receive read-only replicas of the current state. The message frame has control fields for the data replication scheme and payload; these are illustrated in Fig. 3.21 using the Maximum Transmission Unit (MTU) of IEEE 802.15.4. The preamble, SFD, and total packet length together constitute a 6-byte physical layer header followed by **CMD** byte, which is translated into 3-bit for **C**ounter byte, 1-bit for **M**essage type and 4-bit **D**ifferentiator for cluster differentiation. The leader identifier is the 16-bit short address in IEEE 802.15.4 networks [55]. The counter is the

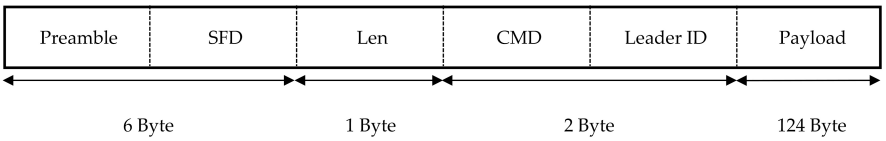

Fig. 3.21 Data replication broadcast packet format [55]

relay counter to decide the number of rounds required for re-transmission (which is limited to 8 hops since time synchronization deteriorates after eight hops [60]). The replicas are byte encoded payload decoded as regular expressions after reception of the packet. The pattern of the replicas are available at the leader: (*i*) 2 byte owner identifier of the replica, (*ii*) 1 byte data type identifier (*iii*) delimiter (00000000). There are seven basic data types available (integer, unsigned integer, float, double, long, character, and Boolean), each of which is represented by the byte equivalent of their first ASCII character (i.e., for **f**loat the representation is 01100110). Message type bit is used to announce changes in the pattern along with the owner of the replicas. This allows for dynamic changes to the pattern of the replicated data during run-time. The leader manages the replica pattern, where nodes publish a described value used to interpret the pattern. The replication is

propagated across the network in three steps. A change is generated by the owner, who has to write to part of the replica. For example, an integer, a, is replicated across the network, with the initial value 0. A *change* is generated due to a physical trigger or state change event. The generated change is propagated to the leader by performing a *write*. This is a point-to-point transaction between the leader and owner, which is acknowledged. Once the change is received, the leader performs a *merge* on the replica at the corresponding index of the 124-byte payload. During the next *broadcast* round, as shown in Fig. 3.24, the change is propagated to the first hop, followed by *retransmit*, where nodes propagate the change to the subsequent hops [55].

This data replication scheme using WSN presents a solution for collaborative systems using the concept of *Decentralized Brains* and reduces the number of wireless transactions required for data propagation and minimizes per node channel occupancy in terms of time and transmit power [55].

In this section, the three main features for the *Decentralized Brains* are discussed. At first network discovery using the periodic transmission followed by a leader election protocol which is very important for a network that has ever changing properties. Finally the data replication protocol in Decentralized Brains, where all of these communication primitives along with the support from the networking described earlier is designed [55].

3.5.4.1 Network Discovery

Network discovery is a function that is critical in networks that do not have a control center that facilitates the association and disassociation to a network. In a decentralized networking architecture, network discovery is the first step in network formation. In this step, the node first joins or creates a network depending on the functions of the node [55]. In this networking architecture, the network takes part in a synchronous broadcast round, to facilitate this behavior, one node in the network is elected as a leader which is detailed in *Section 3.5.4.2*. Usually, synchronous broadcast rounds are only used for data replication, but it can also be used to perform network discovery. When a node turns on, it listens on the channel for the periodic transmission. When it has found the periodic transmission, it joins the network. If a timeout occurs at the discovery state, which is the *discovery timeout*, the node has not found an existing network. This node becomes the leader, and waits for other nodes to join the network while periodically sending a broadcast message until *join timeout* is reached where the node enters the discovery state as shown in Fig. 3.22. In certain cases, this network discovery primitive is also used to segregate clusters of nodes. When a node belongs to a certain cluster, it listens

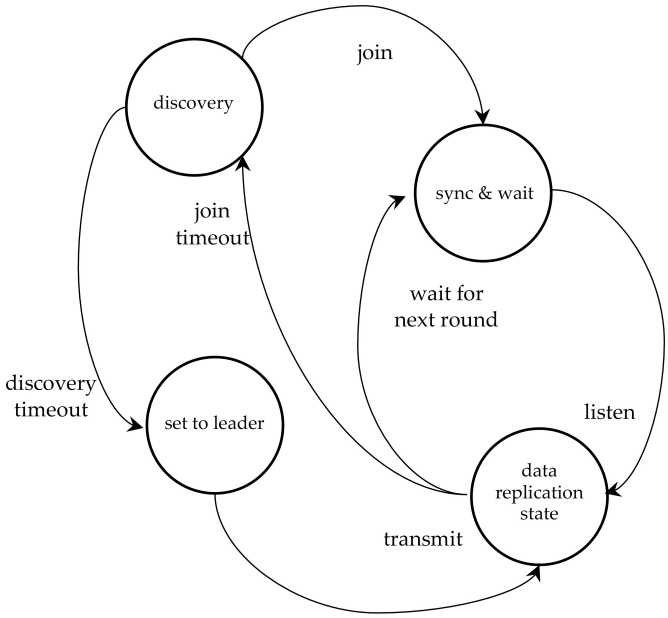

Fig. 3.22 Leader discovery states of new nodes for data structure replication [55]

to a specific periodicity whereas the network performs a different function than the other. This kind of clustering the nodes is required to increase the data throughput of data replication. This can be considered as a publish subscribe model in distributed systems communication. Where the cluster identifier has topics and nodes can discover the topics using this network discovery primitive for selective listening to messages depending on their functions [55].

3.5.4.2 Leader Election

For the distributed data consensus, a leader election process is required. The leader initiates the communication rounds for synchronizing the nodes. Usually, a node in the network that has the most energy is desired to be the leader in the network. In this consensus scenario, the leader is elected using different hop count, address identifier, and user set priority for a node to become a leader in the network. The process of leader election is triggered by a timeout that arises in the network where the nodes are implicitly synchronized. The number of missed rounds is set as the trigger for the election process. The election process always arrives at a leader with consensus from all nodes. Since the data objects replicated by the broadcast

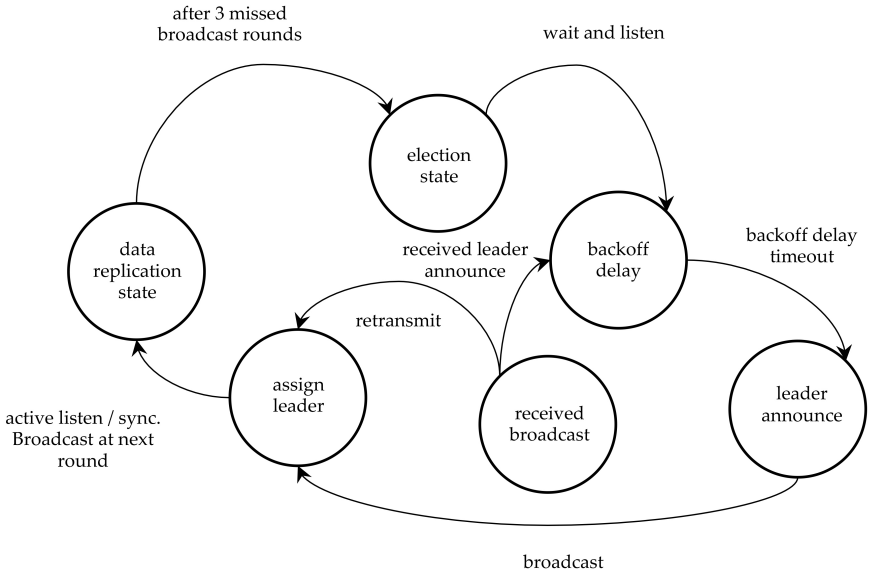

Fig. 3.23 Leader election for distributed data structure replication [55]

require a leader, the termination of the leader election process is very critical. Rather all nodes become stale and wait for a leader to be elected. When the defined number of synchronous broadcast rounds is missed, all the network nodes become active. Since all nodes wake up for the broadcast, all the nodes in the network enter an election state [55]. The election state lasts for a duration from the last missed broadcast round to the upcoming round. This duration is split into consecutive synchronous broadcast rounds, where the initial three rounds are reserved for the nodes with the user set priority. The rest of the slots are used by the nodes randomly or depending on an objective function that minimizes the *back-off delay* [1] of the nodes. In this implementation, a random time slot in the given number of time slots is chosen for sending the *leader announce* message. The node that has the minimum *back-off delay* will start its broadcast depending on the available slot. The node sets the back-off delay timer, and it actively listens for any incoming broadcasts. Suppose the incoming broadcast is the *leader announce* message during the *election state*. In that case, the node will reset the back-off delay timer to a specified delay to achieve synchronous

[1] A random time delay referred to as the back-off delay, has been widely used in collision avoidance strategies during simultaneous transmissions leading to a collision. This MAC protocol is called CSMA with collision avoidance, and its origin of using the back-off delay for this purpose can be traced back to Ethernet. It is also used in the Wi-Fi protocol implementation

transmission, wait and retransmit the received message and finally set the leader identifier to the node identifier received during the broadcast. In this manner, all nodes reset their back-off delay and synchronously retransmit the received broadcast until a counter is reached to get network-wide consensus on the leader election. Moreover, to achieve the next synchronous broadcast round with the continuation of the data replication, the nodes remain in active listening mode to receive the re-synchronized broadcast round from the newly elected leader, as shown in Fig. 3.23. In the best-case scenario, a node with the best user priority will become the leader. It is assumed that the same user priority is not defined for another node during deployment, or the node that chose the earliest slot during the synchronous broadcast becomes the leader. In the worst-case scenario, the nodes end up with a minimum of two leaders in the network, which is an undesired state. It can be mitigated by choosing a better objective function for the back-off delay or reducing the number of nodes participating in the election by filtering the nodes to participate in the election. One strategy for filtering could be with the node address and reducing the probability of two nodes with the random back-off delay [55].

3.5.4.3 Data replication

The data replication primitive is implemented on top of the 6LoWPAN network after the leader election occurs. Leader election is the first step after which the leader discovers all the neighboring nodes in the network that might want to take part in the data replication primitive. The data

	States	Owner	Leader	Hop 1	Hop 2
	Init	a = 0	a = 0	a = 0	a = 0
	Change	a = 1	a = 0	a = 0	a = 0
t_w	Write	a = 1	a = 1	a = 1	a = 0
t_m	Merge	a = 1	a = 1	a = 0	a = 0
	Broadcast	a = 0	a = 0	a = 1	a = 0
t_r	Retransmit	a = 0	a = 0	a = 0	a = 1

Fig. 3.24 Data replication states from source until propagation [55]

replication primitive requires a reliable broadcast mechanism to replicate data. The amount of replicated data is restricted to the MTU frame of the IEEE 802.15.4 to have the nodes coexist with networks that might run other networking protocols. The leader replicates the available payload after the MAC frame headers. The frame consists of a 4-bit counter and a 4-bit cluster identifier. The leader schedules broadcast round a time slot. This time duration is communicated to all the other nodes to wake up reliably for the next round of transmission. After this, the broadcast round is set up in the initial implementation. Every time a payload is sent using synchronous broadcast, the memory is overwritten with the newly received bytes. There are proposals to communicate a dictionary with an objective function for the nodes during the setup phase. The dictionary serves as a map for the data that provides meaning, and the objective function can be used as an operator for the received data. These are two methods with which elaborate collaborative applications can be programmed. The above-proposed data replication is a read-only primitive where the changes have to be written to the leader [55].

The networking paradigm is implemented using Contiki-OS and tested in a CC1350STK. A 6LoWPAN network is deployed in the nodes, where there is a border router. Routes are established using the RPL routing algorithm, which is the de facto standard of routing in lossy and low-power networks. This network establishes IPv6 addressable nodes in the 2.4 GHz band, where the network functions as a multi-hop network. Each node has an IPv6 address, which is reduced to a 64-bit and a 16-bit address. The border router manages the addresses, and packets can be routed outside the network using packet conversion. Therefore, it is easier for heterogeneous networks to interact, such as sensing applications where the sensor collects data in the field. The data can then be warehoused and used for diverse applications on stations that communicate using different means [73]. The data replication is a modular communication primitive that provides easier state management between nodes and is implemented with reliable atomic broadcasts in MSP430-CCRF. The time required per update propagation t_p can be estimated using the sum of individual timing requirements in the networking architecture. As shown in Fig. 3.24, t_w,t_m,t_r are the durations required for effective replica propagation from the owner to all other participating nodes. t_w is required for point-to-point communication between the owner and leader, including acknowledgment. t_m is the time required for the leader to merge the data received. If the leader nodes are resourceful, it is possible to offload computation utilizing merge operators. t_r is the collective time required for one broadcast round with all participating nodes in multiple hops. The update propagation time is given by

$$t_p = t_w + t_m + t_r$$

3.5.5 Decentralized Brains: Implementation and Evaluation

Decentralized Brains focuses on the communication between the nodes in scenarios of self-assembly and heterogeneous collaboration of nodes taking care of the communication payload dissemination. This allows the implementation of hardware-based extensions to existing centralized industrial scenarios to retrofit decentralized collaboration on a network level [55]. Using such a communication paradigm makes it easier to implement and deploy multiple parallel control of decentralized systems. An abstract example from networking and communication is when a node wants to discover and join a network and wants to start operating within the network to perform collaborative tasks. The collaborative task can be for self-assembly or task coordination. The control systems have to exchange precise information between interacting nodes and disseminate the change in global states across all network nodes.

3.5.6 Design of experiment

There are two main differences between the implementation of Glossy [60] and synchronous broadcast in Decentralized Brains [55]. The target hardware where the synchronous broadcast is developed is a dual-band SoC, which can run two radio networks in two different physical layers. We choose the 2.4 GHz for data communication in a multi-hop mesh network topology where the nodes communicate with addresses in a 6LoWPAN IPv6 networking paradigm. The synchronous broadcast is developed in the Sub-1-GHz physical layer. Two main reasons for such a choice in developing the synchronous broadcasts is to increase the range of each of the nodes and to reduce the strict temporal distance window to improve the reliability of synchronous broadcasts. Moreover, it leverages the flexible, hardware-specific simple link SDK for deploying a low-power multi-radio network. We precisely developed the synchronous broadcast in the 868 MHz bands because the temporal distance is higher to achieve constructive interference. It allows the nodes to be synchronized less frequently. Therefore, the precision of synchronization required can be less compared to glossy [60].

Why CC1350? is because of the ability to run two different radio networks in two different physical layers and also that the simple link SDK allows it to be ported to other MCU from TI to enable developers to choose hardware depending on the application scenario freely.

Due to the hardware selection and the development of synchronous broadcast in 868 MHz, it is not necessary to implement the time

synchronization as required in Glossy [60] as the temporal distance required for performing reliable constructive interference increases. Here, implementing the time synchronization using the two hardware clock sources and not using an extra hardware capability to track the clock changes. The functions of a capture unit as the required by [60] are performed inherently by the hardware.

Implementation: Time synchronization is required for synchronous broadcasts to achieve constructive interference. It is implemented in [60], where time synchronization depends on two clock sources, which are the Real-Time clock (RTC) and another high-frequency clock. The high-frequency clock is sourced from the MCU's internal clock. It is widely known that this clock is unreliable due to skew and MCU's operating temperatures. RTC provides better accuracy against clock drifts in the long term, and the higher frequency clock provides better time resolution to achieve accurate synchronous broadcasts. For the implementation in [60] the VHT approach [61] was used, which uses both the RTC and an internal high-frequency Digital Controlled Oscillator (DCO) from MSP430 MCU. VHT ensures high precision in time synchronization and low-power consumption, which has been thought to be an oxymoron in designing distributed low-power electronics [61].

For our implementation, Dual-band CC1350 MCU is used with TI simple link SDK. CC1350 is a dual-band SoC containing 2 ARM Cortex cores. One is the central MCU core that runs the user logic and the operating system, and the second core is dedicated to the radio functions and dual operation. The dedicated radio core is implemented to power up and down depending on the radio usage for power conservation purposes.

CC1350 radio operations are scheduled and time-stamped with a separate 4 MHz timer, called RAT timer. Therefore, radio operations scheduling resolution is limited to that frequency. This dedicated core/timer of CC1350 gives it an advantage of running flooding protocols as it guarantees the exact timing of the execution of radio commands. Hence, it provides more deterministic control over temporal distance and required delays between synchronous broadcast floods.

VHT [61] is used in Glossy, but in the *Decentralized Brains*, we use the hardware supported implicit clock synchronization. This synchronization happens between the 4 MHz RAT timer and the 32 kHz RTC clock. As the RAT-RTC synchronization command is issued for the first time, the radio core waits for the next RTC tick to start the RAT timer. To handle the radio core's power-downs, every time before issuing the power down command, a RAT stop command is issued. This command returns a synchronization parameter passed to the RAT start command at the next radio core power up. This method keeps consistency between RTC and RAT and preserves the required clock resolution and power limits.

Another feature of CC1350 is that time stamping of received packets is done automatically by the RF core using RAT time. These two features simplify the calculations of time for the next flood and provide a deterministic behavior of the nodes' temporal distance. Instead of a software time stamping, calculating, and waiting for a software-delayed time as in [60], *Decentralized Brains* implements a full hardware-based approach on CC1350. To adhere to the requirements for low-power operational constraints, the *Decentralized Brains* software uses only event-based callbacks for scheduling of floods and other auxiliary run-time software components.

As soon as a non-initiator node receives its first flood packet, it stores its base time with the received packet time stamp, increments the packet relay counter, and then transfers to transmission mode. The time between packets within a flood is fixed, so the node immediately issues a transmission command to the RF core to send after this fixed time. The RF core handles the power up and down and automatically wakes up before the transmission time. Depending on the specified number of transmissions, the node may switch to the receiving mode until the re-transmissions are done. Also, as the time between floods is fixed, the node schedules the next flood (the first receiving command of the next flood) with each flood's last callback.

3.5.7 Why use Contiki-NG?

The Contiki-OS official documentation states the following and delivers on those points, which is the primary motivation for using Contiki-NG.

> Contiki-NG is an open-source, cross-platform operating system for Next-Generation IoT devices. It focuses on dependable (secure and reliable) low-power communication and standard protocols, such as /6LoWPAN, 6TiSCH, RPL, and CoAP. Contiki-NG comes with extensive documentation, tutorials, a road-map, release cycle, and well-defined development flow for smooth integration of community contributions.

6LoWPAN is a networking standard implemented in the Contiki-OS used in the decentralized networking architecture for point-to-point communication with the leader for writing changes. BLE is a low energy protocol, but it does not provide all of the networking features. Even though the BLE beacons are highly scalable and require significantly less energy, they do not provide scalability when the number of nodes increases in point-to-point communication in any networking topology. ZigBee is another standard that is widely used in connected home architecture, and the latest standard specifications of ZigBee provide IPv6 support. It has self-healing and network discovery features that are required for a

decentralized network. Since the networking standard was strictly developed for home automation standard, it does not offer a developer the flexibility to easily adapt certain networking layers or the MAC layer. Therefore, Contiki-OS, with its implementation of 6LoWPAN is chosen as the network that drives the nodes' primary networking.

Above all of this, it also liberates the developer and provides an operating system that is hardware agnostic. We strive to develop the *Decentralized Brains* with the same philosophy of modularity in code-base with platform agnostic development approach as in the case of Contiki-OS. This approach enables the developers to apply synchronous broadcast communication primitive however possible in applications and not only for distributed consensus in low-power WSN. Moreover, the software support with well mature features required for developing the *Decentralized Brains* networking layer is provided by Contiki-OS. This allows for further extending the *Decentralized Brains* networking paradigm into other hardware systems and applications. The network stack of Contiki-OS is shown in Table 3.1, which shows the necessary components developed for the IEEE 802.15.4 standard. It provides the MAC layer for 6LoWPAN networking using CSMA /CA. The operating system also provides the flexibility to turn the features on and off during run-time, which requires developing the synchronous broadcasts communication primitive. Here the MAC layer is extended to provide synchronous broadcasts as the opposite of listen-before talk is required during this type of network flooding. In this network flooding, all nodes that will participate in the synchronous broadcast round will not listen to the channel, instead transmit the payload as soon as the time constraints are met.

OSI layers	Contiki-OS	*Decentralized Brains* Implementation
Application	web-socket, http-socket, coap.c	DeBr
Transport	udp-socket, tcp-socket	
Network, Routing	uip6, rpl	not required
Adaption	sicslowpan.c	
MAC	csma.c	time synchronised scheduling
Duty Cycling	nullrdc, contikimac	synchronous broadcast
Radio	MSP430, CC1350, ...	CC1350 (Sub-1-Ghz)

Table 3.1 Contiki netstack along with *Decentralized Brains* netstack

With a strict time synchronization protocol, an effect called constructive interference, as described in Sec. 3.5.1.1, is achieved between the nodes that are simultaneously transmitting in the medium [55, 60]. The Radio Duty Cycling (RDC) layer and the MAC layer are the two layers that require flexible and scheduled operation using the synchronized time. The modularity provided by Contiki-OS in such granularity makes it easier to

implement features necessary for *Decentralized Brains*. A reproducible, modular code base that can be used to develop applications is the goal of *Decentralized Brains*, and Contiki-NG accelerates this process.

3.5.7.1 6LoWPAN in Contiki

The right half of Table 3.1 shows the different layers in the Contiki-OS netstack. The most straightforward layer in the IoT/IP stack is the MAC layer. This layer is used to avoid collisions using an exponential back-off delay if there is traffic after probing the channel for any transmissions from other nodes. This probing to measure the spectral energy is above or below a threshold called the CCA.

The MAC protocol used is called the CSMA/CA. The CSMA/CA works by sensing the medium before transmitting to acquire CCA. The CCA, when positive, is followed by a transmission. When the CCA is false, the node will wait with a back-off delay to ensure that the medium is clear for the next transmission attempt. Contiki-OS provides out of the box networking layer for 6LoWPAN which automatically forms a wireless IPv6 network with the routing protocol called Routing Protocol for Low-power and Lossy Networks (RPL) network layer, which is part of the network layer as shown in Table 3.1. Along with header compression and segmentation, the large frames can be accommodated in IEEE 802.15.4 limited packet size. The network layer contains two sub-layers, the upper IPv6 layer and the lower adaption layer, which does the conversions. The Adaptation Layer provides IPv6, UDP header compression and fragmentation to transport IPv6 packets with a Maximum Transmission Unit (MTU) of 1280 bytes over IEEE 802.15.4 with an MTU of 127 bytes. In Contiki-OS implementation, the routing protocol is called ripple. RPL forms a routing graph from root nodes or an access point, which is a *de facto* routing protocol for this class of networking. It builds an acyclic graph from root nodes called Destination Oriented Directed Acyclic Graph (DODAG) [74].

There are two kinds of routing modes called the storing and non-storing mode. The non-storing mode does not store the routing information in each node, reducing the memory strain. Highly unstable networks must use the storing mode since it can heal faster upon failures than the former. A multi-hop mesh network is created using the netstack (network stack) provided by the network. These features are used as the underlying communication layer that transports the data from the data originator that sends the data to the leader node which propagates the data to the rest of the nodes for data replication.

3.5.8 Experimental setup and evaluation

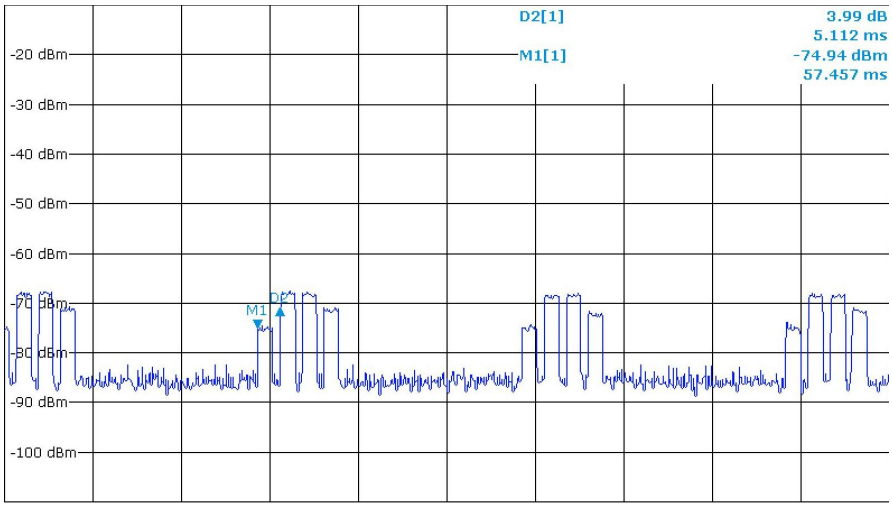

Fig. 3.25 Received signal power measured by spectrum analyzer. The X-axis is a time sweep of 2 s with distance between two vertical lines measuring 20 ms. The experiment is performed using one initiator node and one non-initiator node.

This section presents the following three parts: *(i)* developing and performing experiments to prove the effect of concurrent transmissions, *(ii)* a large-scale experiment developed to analyze the performance of synchronous broadcasts and finally *(iii)* the performance analysis to understand the reliability of synchronous broadcasts used in *Decentralized Brains* for developing decentralized consensus algorithms. Initially, the effect of concurrent transmission on received signal power and distortion is presented with measurements using a spectrum analyzer. As discussed in Sec. 3.5.3, multiple nodes are transmitting simultaneously within an allowed window of time. This experiment visualizes the received signal power measured with a spectrum analyzer listening to 868 MHz, where nodes are placed away from the measuring antenna equidistantly.

Observing effect of concurrent transmissions: Fig. 3.25 shows the effect of concurrent transmission when there are two nodes, which are the initiator node and the non-initiator node. In Fig. 3.26, the experiment with one initiator and 3 non-initiator nodes transmitting simultaneously with the allowed window of temporal displacement is shown. In both experiments, including two nodes and four nodes, the nodes receiving the packets from the designated initiator node re-transmit the packet three times as soon as

they receive it. The number of re-transmission is arbitrarily chosen as three to demonstrate constructive interference during synchronous broadcasts.

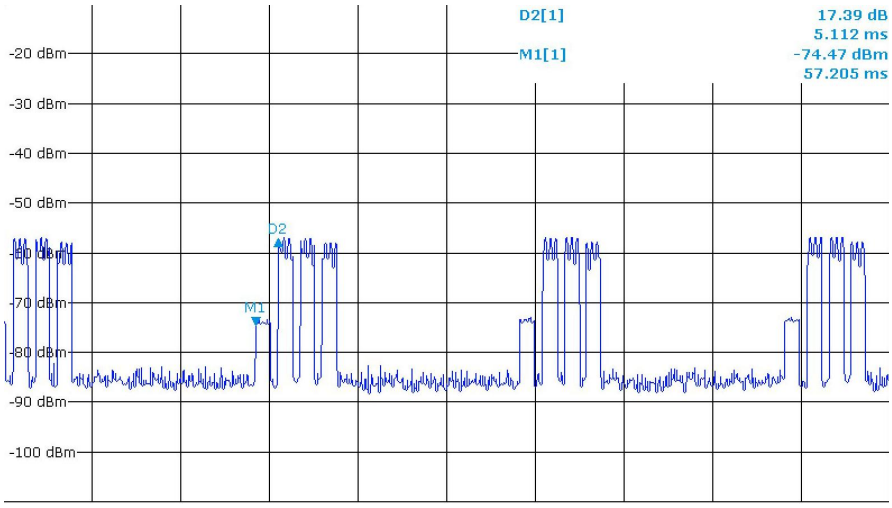

Fig. 3.26 Received signal power measured by spectrum analyzer. The X-axis is a time sweep of 2 s with distance between two vertical lines measuring 20 ms. The experiment is done using one initiator node and 3 non-initiator node.

It can be observed from Figs. 3.25 and 3.26 that every time a transmission occurs, there are peaks in the signal acquisition, which are marked by the markers *M1* and *D2*. During these transmissions, since there is more than one transmitting node, the received signal power increases at the receiver as well as signal distortion increases with the number of concurrent transmissions. The first peak with the marker M1 is a packet transmitted by the initiator node; hence, the measurement is not distorted, and the received power is lower compared to the following peaks. The second and third peaks in the figures come from all of the nodes. Hence, they have the highest power and highest distortion. The last packet in each flood comes from non-initiator nodes. Hence they have received a measured power slightly smaller than the previously received packets' measured power. This experiment is repeated for multiple rounds to ensure the effect of constructive interference.

In the *Experimental setup*, we present two experiments where both of the experiments were performed with a physical layer radio configuration of 50 Kbps in a 2-GFSK modulation. The first experiment empirically quantifies the allowed temporal distance. This effect was already demonstrated for 2.4 GHz using the QPSK modulation in Glossy [60], but we experiment with a Sub 1-GHz band to prove the feasibility of constructive interference in this frequency band. Temporal distance is the allowed time window within

which the effect of interference is constructive. The second experiment provides an insight into the effect of constructive interference due to the number of concurrent transmissions. It also helps in studying the effect of constructive interference in relation to the number of neighbors transmitting simultaneously.

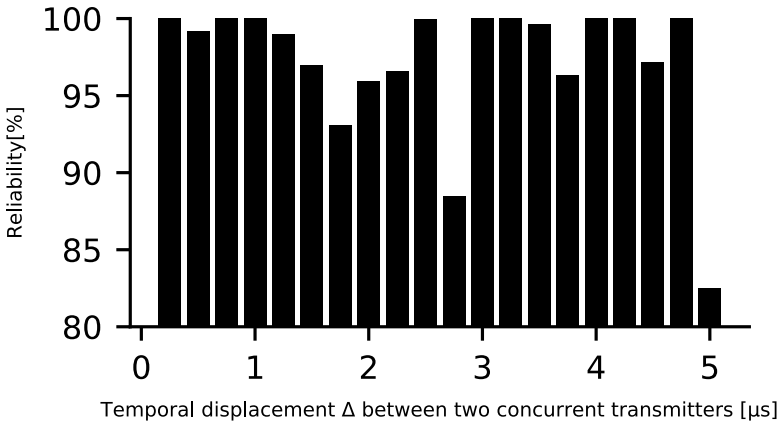

Fig. 3.27 Two nodes transmitting with different delays starting at 0 where absolute concurrent transmission occurs. The x-axis is the delay with a resolution of 0.25 μs. The experiment is performed for 20 times starting from 0.25 μs delay to 5μs delay.

The first experiment is performed to prove that constructive interference works for Sub 1-GHz and calculates the minimum temporal distance. The experiment is performed by sending a signal from two nodes, one is delayed, and the other is non-delayed. To prove constructive interference and prevent the capture effect from making the receiving (RX) nodes capturing the first signal, the non-delayed node sends at a power of -10 dBm. The delayed signal transmits at a power of 0 dBm. The experiment is repeated multiple times, increasing the delay by a step of 0.25 μs each time. As observed in Fig. 3.27, the Sub-1-GHz behaves in a similar way to 2.4 GHz [60], where the shift in temporal distance results in several valleys and peaks. The signals get constructively and destructively interfered, but Sub-1-GHz (2-GFSK) seems to be prone to bigger temporal distances than 2.4 MHz as demonstrated in Glossy [60]. Fig. 3.27 shows that the allowed temporal distance for Sub 1-GHz to perform synchronous broadcasts due to constructive interference reliably can be 1.25 μs. The signals can constructively interfere with a reliability of 97.06%.

The second experiment studies the effect of a number of neighboring nodes during synchronous broadcast rounds. The experiment starts with two nodes transmitting simultaneously. With each round of synchronous broadcast, the number of participating nodes is increased with an extra node. As observed in Fig. 3.28, flooding works at a 99.99% when there are 2 to 3 neighboring nodes. The behavior starts to deteriorate in the experiment as more nodes join the synchronous broadcast round. After three nodes with high reliability, the nature of interference slowly shifts, reaching reliability of 72% when there are 8 neighboring nodes. Even though the reliability is above 70% for 8 nodes, for every 10 packets sent, the constructive interference effect cannot be seen, and three packets are lost. With a long spatial stretching, a densely populated network for distributed sensing, or collaborating control systems, this reliability can still offer specific capabilities that were not possible previously. The delay between the flooding rounds can be tuned depending upon the application requirements. The frequency of the flooding initiated by the initiator and the time between each node's re-transmission can be configured with the pre-determined software delay.

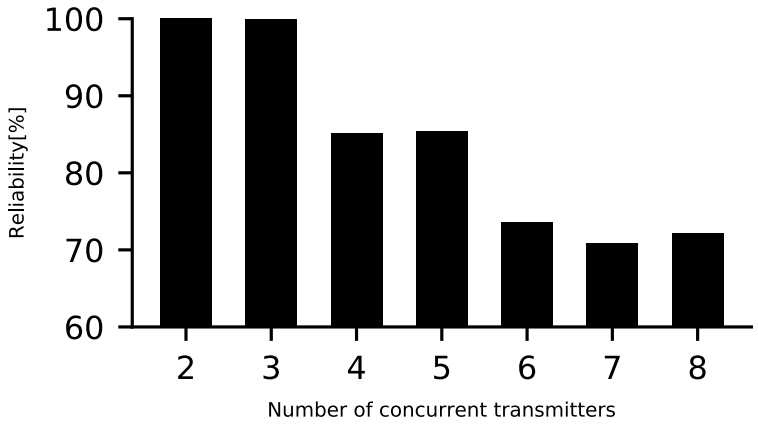

Fig. 3.28 The effect of the number of neighbors on the reliability of the packet received. The number of concurrent transmitters starts at 2 and increased by 1 for each next experiment.

Performance evaluation: In this section, we test our implementation on a local testbed (Sensor Floor). We use a 345 node deployment to test the implementation. The 345 node deployment is called the Sensor Floor, a sensor network array embedded under the floor with CC1350 sensor tags. The floor contains 345 nodes. A subset of 101 nodes randomly for every

experiment runs to count for the effect of spatial orientation and environment of the nodes for multi-path and scattering effects. 1 Initiator and 10 nodes for 10 concurrent transmissions are chosen randomly from the network of 345 nodes. Additionally, these random patterns will allow testing against different pattern arrangements of nodes and the distance between nodes in the range of 1 to 30 meters. The pattern choice and the sequence of transmissions performed for analyzing the characteristics of constructive interference is listed in Table 3.2.

Node/re-transmission	0	1	2	3	4	5	6	7	8	9	10	11	12
initiator	TX	RX	TX										
sequence 0	RX	TX	RX	TX									
sequence 1	x	RX	TX	RX	TX								
sequence 2	x	x	RX	TX	RX	TX							
sequence 3	x	x	x	RX	TX	RX	TX						
sequence 4	x	x	x	x	RX	TX	RX	TX					
sequence 5	x	x	x	x	x	RX	TX	RX	TX				
sequence 6	x	x	x	x	x	x	RX	TX	RX	TX			
sequence 7	x	x	x	x	x	x	x	RX	TX	RX	TX		
sequence 8	x	x	x	x	x	x	x	x	RX	TX	RX	TX	
sequence 9	x	x	x	x	x	x	x	x	x	RX	TX	RX	TX

Table 3.2 Testing on the Sensor Floor for re-transmissions (N=2). For each of the 10 sequences, 10 nodes transmit concurrently. Each sequence is programmed to discard packets less than their sequence number based on the counter byte in the packet (First byte)

As the Sub-1-GHz band can operate over greater distances compared to our testbed area, we simulate the effect of hops as in the case of spatially distributed nodes in our experiments. Flood packets have their first byte used as a counter, this counter is set to zero by the initiator and incremented by 1 by all nodes with each re-transmission within a flood. Nodes are split into sequence groups, each group discards packets with a counter number less than a specific programmed number. As shown in table 3.2, where the tests are presented, when the nodes are programmed to do 2 re-transmission (N=2), with discarded packets marked as (x). Each sequence group keeps discarding flood packets until their programmed sequence number. When the sequence group identifier matches the counter byte, then the sequence group joins the flood. In this manner, a multi-hop test of 10 hops is simulated in our testbed for evaluating the performance of spatially distributed nodes.

A flood is considered successful when a node receives the flood packet with the correct CRC (cyclic redundancy check) at least once, but we do not count floods if no packets are received. The success rate is the rate of

successful floods over the overall number of floods averaged among all nodes on the floor. Each of the tests runs for 50000 packets.

Table 3.3 shows the result for the different number of re-transmissions. As can be observed, the increase of re-transmissions helps in increasing the success rate. The success rate can reach 99.50% when there are 6 re-transmissions per synchronous broadcast flood round.

Concurrent transmissions (N)	Average success rate	Max no. of concurrent transmissions
N=1	98.72%	10
N=2	97.51%	20
N=3	97.94%	30
N=6	99.50%	50

Table 3.3 Overall success rate of testing on Sensor Floor

3.5.9 Energy Profile of Decentralized Brains

The energy profile shows the energy requirements at different stages of the state machine execution. Here, the synchronous broadcast is profiled to show the effectiveness of energy usage. The most costly action on a wireless sensor node is a transmission. Since synchronous broadcast relies on transmissions, it is essential to show the required energy during a synchronous broadcast. A device that is not the initiator of the synchronous broadcasts is chosen to profile energy during the synchronous rounds. The justification for choosing an end node is to understand the field devices and the requirement of energy when deployed with a synchronous broadcast paradigm in IoT based use cases since the initiator is a device with abundant energy to operate actively for long periods. The power source is a digital power supply, and an initiator node is performing synchronous broadcasts where the end nodes are participating. Detailed waveform analysis of the energy requirements is discussed below with the following six bullet points, each of which corresponds to 6 recorded peaks in Fig. 3.29.

1. Startup phase or boot phase from the low energy mode to receive mode. Here initialization of all timers, clock synchronization, and the timers to extrapolate the high-speed clock from the slow clock are performed.
2. The second peak is the reception of the first message from the initiator during the synchronous broadcast round.
3. Until the third transient, the observed current draw, is the pre-defined software delay to synchronize all transmitters after message timestamping. The third peak is a decentralized synchronous transmission, where all

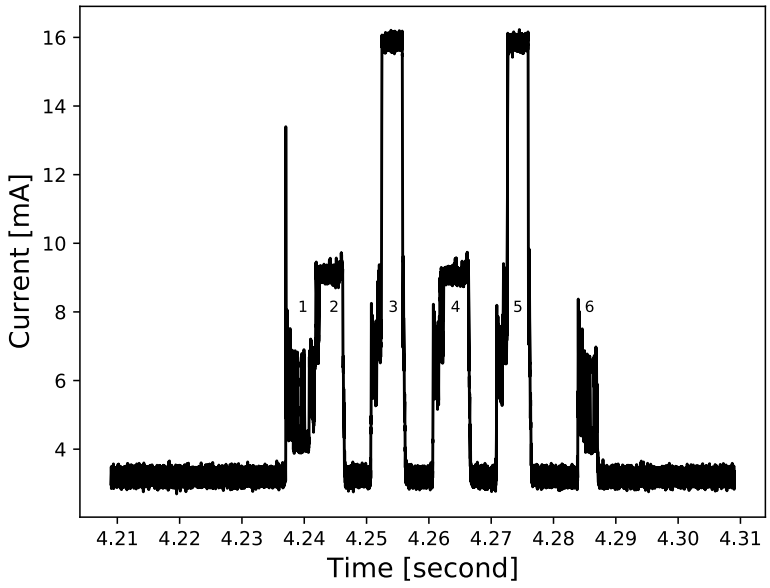

Fig. 3.29 Energy profile of synchronous broadcast

other nodes are expected to be within the guaranteed temporal distance for effective constructive interference during transmission. The third peak observed in the figure is higher than the previous peak as transmission requires more energy than reception.

4. After taking part in the first transmission round, the node stays awake, listening to the next message to be transmitted again. There are four synchronous transmissions. The device under investigation will send only half the time as the alternating times; the node will listen for the same payload and clock synchronization.
5. The fifth peak is the alternating second synchronous transmission attempt by this node. It is crucial to create an optimization plan across all nodes on the number of times a node can participate in synchronous broadcast transmission. In this scenario, a round is with four attempts, and every node can only participate two times. The larger and denser the network, the number of transmissions within a flood round should be increased. Reducing the participating nodes per round will optimize the network's overall energy cost.
6. The sixth peak is the final round, where the node schedules its next network flood round to wake up if the application wants to conserve energy or allows other application-based processing and communication.

The energy required is relatively lower than the reception and transmission since the CPU schedules a timer enabling all the interrupts and returns to idle in this experiment scenario.

Cumulatively calculating the energy requirements from the measured energy profile, there are four synchronous broadcast rounds. We can calculate the overhead for decentralized synchronization as two transmissions and two receptions, each of which requires 13.4 mA at 10 dBm and 5.4 mA, respectively, as per the data sheet [51]. Use case dictates the frequency of decentralized synchronization that limits the recurrence of the synchronous broadcast. We can plan broadcast rounds with up to ten minutes of time delay between each round. This is the allowable limitation for time synchronization using message time stamping, after which the nodes cannot guarantee the temporal distance for synchronous broadcasts. Therefore, we can achieve decentralized synchronization at its highest level in the low-power wireless sensor networks with its calculated energy overhead required to perform the necessary communication primitives.

3.6 Conclusion

The *Decentralized Brains* communication paradigm has been conceptualized, designed, and developed in this part of the work. The designed protocol is implemented in a dual-band CC1350 SoC, enabling various other networking applications in the IEEE 802.15.4 standards. This allows for an open implementation and adoption of the synchronous broadcast communication primitive in low-power, low data-rate applications. The performance analysis of the synchronous broadcast has been presented in Sec. 3.5.8, which was performed in the logistics warehouse research facility. A packet success rate of 99.5% was achieved with a 345 node transmitting 50000 packets within the network. Increasing the concurrent transmissions for flooding rounds reduces the packet success rate but increasing the number of re-transmission will allow for mitigating this effect. Stringent time synchronization will allow for better results with calibration for static nodes to estimate the packet propagation time between the nodes.

Low-power, low data-rate wireless communication techniques are of inherent interest for space applications. The integration of decentralized network discovery, dynamic leader election that terminates with a guarantee, and reliable data replication within 3 ms across hundreds of nodes facilitates swarm behavior applicable for industrial scenarios and space applications, which are explored in 5.2 and 5.1. This three-part network communication integration is the main contribution of this work. Complemented with multi-hop 6LoWPAN routing topology, we explore the

stages needed for multi-unit, shared-state maneuvers in the low-power, low data-rate applications. A multi-hop routing is used to ensure network stability and ensure a communication path; as nodes are mobile, it is critical for routes to change depending on link availability. *Decentralized Brains* is a framework for developing distributed sensing and collaborative control systems in low-power, low data-rate WSN networks [55]. The most crucial feature of *Decentralized Brains* called the synchronous broadcasts using constructive interference is developed, deployed in 345 low-power nodes, and tested for performance. Various design choices and differences to existing implementations of constructive interference are discussed. The synchronous broadcasts module for *Decentralized Brains* is developed as part of the Contiki-OS to improve the transferable nature of the concepts and make the features accessible for other industrial applications developers. Leader election and network discovery are two further modules that leverage the synchronous broadcasts to create a decentralized network. When the network loses the initiator node, which acts as the leader node due to energy constraints or because the network is highly mobile, the network enters into a leader election phase where all nodes decide on another leader node using the same synchronous broadcast communication primitive [55].

Porting implementation to 2.4 MHz: As simple link SDK provides a unified API across several TI MCU, our implementation can also be ported to 2.4 MHz devices that use the same SDK, for example, TI CC2650. As per [60], the temporal distance for 2.4 MHz is 0.5 μs, which is possible to be achieved with CC2650. As the radio clock of CC2650 is also 4 MHz, the minimum allowed temporal distance is 0.25 μs. The SDK API for both CC1350 and CC2650 is identical. Therefore the effort to port the code would only involve changes to the RF settings and time between the synchronous broadcast network floods. This will allow for running the currently developed *Decentralized Brains* synchronous broadcast mechanism in CC2560.

Development of a full stack: As TI simple link SDK can run multiple radio instances as part of a single MCU core, we leverage it to implement the synchronous broadcasts. It also allows us to develop a network abstraction to run multiple radio protocols for communication. To make this possible, to improve the reproducibility of the results and to increase the use of constructive interference for network flooding in industrial applications, it is proposed to write dedicated MAC and NET layers in Contiki-OS for the target hardware. Since the target hardware SDK already supports multiple low-power MCU, the same code base can be compiled and reused out-of-the-box. This would allow making the communication stack more code friendly and application realistic.

DezCom:
High-power, high data-rate systems

Industry 4.0 and the concept of a socially networked industry define that the entities in an industry are networked and are able to communicate between each other to autonomously collaborate for performing tasks [75]. DezCom is an architecture for **de**centrali**z**ed **com**munication and messaging. In this work, it is considered that in a decentralized industrial process there are entities communicating, negotiating and coordinating to execute tasks. Even if the existing systems are not decentralized, the underlying communication system is decentralized. The hypothesis of decentralized communication is that any industrial system, when connected, uses a decentralized communication platform where the messages are communicated with the advantages of a decentralized networking paradigm. Even though the execution algorithm is decentralized in nature, when running on a centralized communication architecture, such as MQTT or Orion context broker, it is not truly decentralized. Therefore, the effort to abstract the communication layer with a pure decentralized communication framework is implemented and presented in this chapter as the DezCom communication stack. To prove the hypothesis, minimal changes are performed in a multi-robot system and implemented to perform messaging and accomplish a task.

The decentralization aspect of the communication is developed by creating system specifications based on already existing widely used communication systems and selecting the best state of the art fault-tolerant distributed communication algorithm. Fault-tolerance is provided by blockchain where a byzantine fault-tolerance based consensus is implemented among the communicating nodes. Decentralization using blockchain not only increases the availability of the broker but also increases the security of data in a trust-less network. There are a few preliminary research works available in a literature survey of decentralized robots based on blockchain [76, 77]. In this chapter, we contribute to the research by developing a deployable decentralized industrial supply chain messaging system based on mining-less blockchain. The emphasis on the development is given to messaging using assets i.e., entities in a socially networked industry. This stack described in chapter 4 is demonstrated in the proof-of-concept deployment in Sec. 4.11. Finally, in Sec. 3.6, the outlooks for DezCom are summarized with production notes for reproducing the results for future industrial deployment.

Chapter 4
DezCom: High-power, High Data-rate Systems

Deployment is the action of bringing resources into effective action.
- From the Lexicon

4.1 What is a Context Broker?

Definition 4.1 A *Context Broker* acquires contextual information from heterogeneous sources and fuses them into a coherent model that is then shared with computing entities in the space [78].

Even though the above-given definition was coined in the context of smart homes and automation, the definition is still relevant for industrial context brokers. Then arises the question of what kind of information pertains to a context that is defined by the following from [79].

Definition 4.2 *Context* is any information that can be used to characterize the situation of a person or a computing entity [79].

In a networked industry, every person and entity generates data that can be used to understand the context. Location information forms the basis for a context in an industrial network. There are multi-modal approaches for the precise localization of industrial entities. Sometimes, location and other system information precisely characterize the context. Previous works have pointed out why location information is an essential aspect of context information [80–83], a holistic understanding of context should also include information that describes system capabilities, services offered and sought [78]. Furthermore, context information also includes the activities in which people and computing entities are engaged for collaborative task execution, the spatial and temporal properties associated with the tasks, and their situational and access roles, the objective for optimization, and intentions with which the entities are deployed in the industrial network [78].

Several communication architectures for context-aware computation and task execution have been proposed and developed in the past [11, 78, 79, 84, 85] that understand the context and build situational awareness of the

entities. Although the literature is from another application domain (smart homes, intelligent environments), the transfer and application of the term and its underlying meaning of *context* in an industrial environment are relevant since *context* is an intrinsic characteristic for any collaborating organization of entities in a network. Previous research and literature on building the earlier architectures have made progress in various aspects of embedding computational capability or in the trend of IoT. For example [79] developed a middle-aware framework to facilitate context acquisition, defined new extensible programming libraries for building intelligent room agents [84], and created badge-size tracking devices for determining people's location in an indoor environment [85, 86].

Many of these systems try to propose a solution for the most critical aspect of context-aware computing but fail to meet the requirements or cater to industrial standards to be deployed for long-term operation. Few of these shortcomings are weak support for knowledge sharing between the entities across heterogeneous networks and developers for iterative and version based updates. Lack of adequate user privacy, i.e., the data generated by an industrial system, belongs to the system developer or the deployer of the system and the underlying mechanisms to critically control data access. Interoperability is another vital aspect of context-aware computing and the underlying communication brokers that were not foreseen during those systems' development. In the following examples, the implications of the lack of interoperability are evident i.e., the Context Toolkit framework [79], Schilit's context-aware architecture [85], and the Active Badge system [86]. Context knowledge is embedded in programming objects (e.g., Java classes) that are often inadequate for supporting knowledge sharing and data fusion operations [78].

Fig. 4.1 illustrates the role of a context broker in an industrial scenario. As seen in Fig. 4.1, the context broker becomes the central entity that provides the necessary communication between the collaborating entities. All of the connected systems in the illustrated scenario exchange their context through the broker. Each connected system, agents, and devices use their means of interfaces as limited by their design. For example, a low-power wireless sensor, a context-aware device, would choose to communicate using an event-based (publish-subscribe) communication modality. Context-aware agents are software systems that are the most highly available systems that can communicate using an API. These systems provide different interfaces and tools for communication within the infrastructure and also with the context broker. A highly available system exists when there are no resource constraints in terms of energy and communication bandwidth. A typical highly available system is a software service deployed in a data center where it is always running with guaranteed uptime and available bandwidth. The context broker should

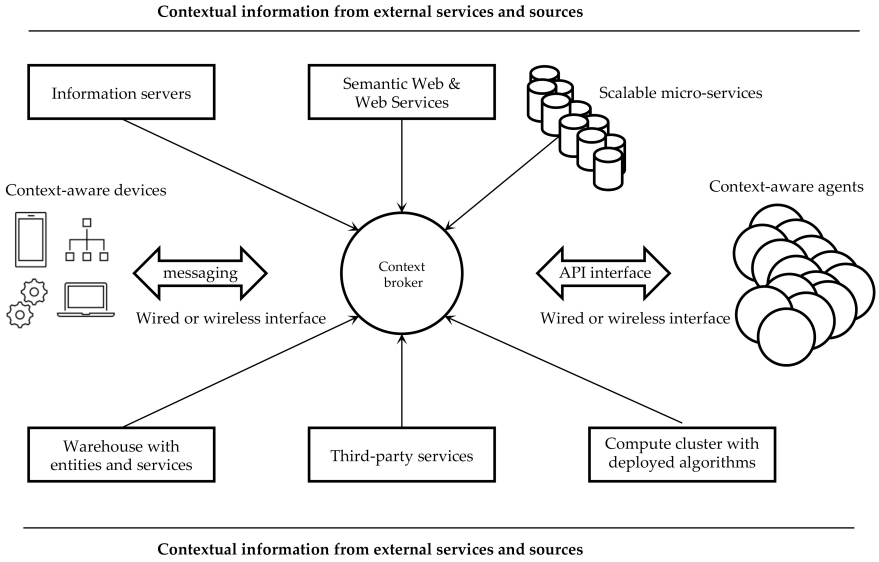

Fig. 4.1 An overview of a context broker deployed in an Industrial application

inherently be able to handle the communication requirements of both of the system types. Even though the software systems and physical sensors have two different ranges in the communication requirement, the system should cater to all of those in the spectrum. To deliver the required quality of service, it is also required that the context broker itself is designed and deployed as a highly available system. It needs to handle spurious event based data from low-power sensors that arise at random where every message sent is more valuable than a highly available software agent. The reason for the messages from low-power sensors to be considered valuable is the cost of transmitting through a network. Every wireless transmission attempt requires energy that can be used to prolong the field usability of the sensor. A low-power sensor in the field communicates to update the context when there is a change. Since they are battery-operated, every message sent must be treated with a higher priority to process and relay the context to the rest of the system. The data is used for further planning and execution of various business processes. In contrast to low-power systems, highly available systems are not as resource-constrained (energy, network availability), so considering the messages from low-power sensors is more valuable and providing a higher quality of service for them.

Two context brokers are considered essential for the study of context brokers since they are widely used in various industrial applications and research projects. In the following sections, the two context brokers understand the working principles and their functionality for industrial

applications. Consider a spectrum of features for context brokers where the lower end is equipped with minimum features, and the higher end is feature full that provides different modalities for communication. The lower end of the feature spectrum encompasses the bare minimum to establish a connection and communicate painlessly. It also does so without increasing the overhead on deploying and integrating the context broker in an existing application stack. On the other side of the spectrum is a complex implementation that provides all necessary features like plugins and requires overhead to set up the context broker itself. We choose to study the two ends of the spectrum to understand a context broker's requirements and the pains and gains in having various features required for integrating it into a decentralized industrial application. On the lower end of the spectrum is the MQTT context broker, and on the higher end is the Fiware Orion context broker. They are widely used in various industrial and retail applications, even though each of these systems' economic impact is not the same; they are widely sought for in applications such as robotics, smart home, industrial automation, and smart city.

MQTT is the first kind of context broker at the lower end of the spectrum that is widely used in many different domains, including highly critical applications such as oil and gas automation industries. There are multiple implementations of MQTT that have the core features and improve MQTT limitations, such as nats.io and ZeroMQ. Due to the study's simplicity and understanding of a decentralized context broker's development, we will deep dive into MQTT and list the possibilities that the other related systems provide. Even though those implementations are available, they are not used as widely as MQTT. This is due to its simplicity in deploying a broker and integrating the service into the application stack. The implementation's programming effort is also very minimal compared to many other featureful context brokers with certain exceptions (ZeroMQ).

The other type of context broker in the higher end of the spectrum is *Fiware Orion context broker*. It is used as an example for this study due to its position in the spectrum. It is a complex system that can handle various communication modalities and plugins, providing different functionalities that can be used for preprocessing on context or post-processing of data before it is committed into persistent storage. An immediate difference between the systems chosen in this thesis is the flexibility in communication modalities and the complexity of the context broker stack itself. In the following two sections 4.3 and 4.4, the Fiware Orion context broker and the MQTT context broker respectively are studied in detail. The study also includes the following *(i)* features it provides for the developer of an industrial application, *(ii)* deployment of the stack and requirements, *(iii)* communication paradigms available, *(iv)*nature and type of flexibility

provided, *(v)* limitations in relation to industrial applications and *(vi)* various performance metrics that are available in literature.

4.2 Context Broker Architecture Standards from ETSI

In this Sec. 4.2, the three different types of systems architecture are discussed to clearly understand the choice of system for the context broker being developed. Furthermore, considering the NGSI standard from the European Telecommunications Standards Institute (ETSI) [87], it is clear that this kind of context broker has not been developed. DezCom is a decentralized context broker with a Byzantine Fault-tolerant consensus system. In the following Sec. 4.2.1, the different types of systems architecture and architectural considerations from ETSI are discussed. This provides a more in-depth insight into the developed system and the exact differences from the existing systems.

4.2.1 NGSI-LD Architectural considerations

The NGSI-LD API is primarily intended as an API and does not set a particular architecture. The NGSI-LD API can be used for various architectural environments, and the design limitations of the API are minimized. Since the NGSI-LD API cannot be used in all the architectures, three sample architectures are provided. The NGSI-LD API aims to enable all of these prototypes to be supported effectively, i.e., design choices for the NGSI-LD API shall be taken into account. The NGSI-LD API can map to one specific device architecture, take elements from different architectures, and combine all prototypes. [87]

4.2.1.1 Centralized architecture

Fig. 4.2 indicates a centralized context broker architecture. There is a Central Broker at the core, which keeps all contextual information. Context producers are using updating operations to update context information within the central broker and context. Consumers use a one-time synchronous query or asynchronous subscribe / publish notification operations to request context information from the central broker. The Central Broker responds to all storage requests. A part that acts as both Context Producer and Context User is presented in Fig. 4.2. The general theory is that components may have many

functions and are not directly stated in the NGSI-LD API standard specification. [87]

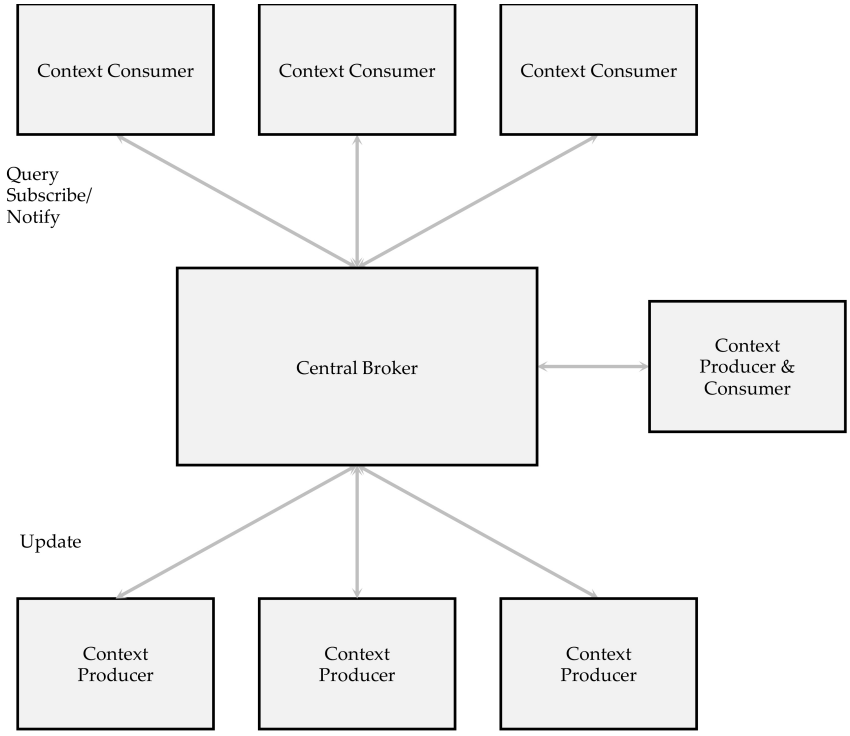

Fig. 4.2 Centralized architecture [87]

4.2.1.2 Distributed architecture

The distributed architecture is presented in Fig. 4.3. The underlying idea is that the Context Sources store all information. As a Context Broker, Context Sources enforce the NGSI-LD API query and subscription components. They register with the context register, provide descriptions of the context. However, they do not update data concerning the context, e.g., some context sources register to provide the indoor air quality of Building A and B or speed up the cars in a geographical area covering the middle of a city. [87]

Context Consumers can require a Distributed Broker to subscribe to it. A node can subscribe to the discovery service from the Registry for specific Context Source, i.e., those that can provide contextual information related to the corresponding Context User request, are identified or discovered on

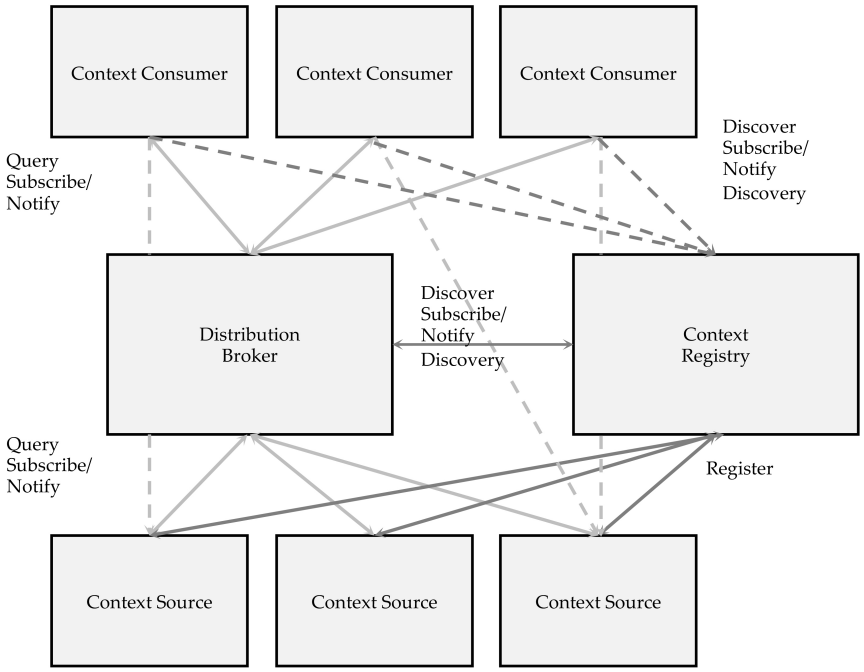

Fig. 4.3 Distributed architecture [87]

each query. The distributed broker will then request or subscribe to each specific context source and add the context information from the context sources. The Distributed Broker can send it to the context user, spatio-temporally, as far as possible. Whether the broker is a central broker or a distributed broker is not apparent to the context-consuming device or user in this service mode. The architecture alternatively enables context consumers to discover context sources by themselves through the Registry and then request them directly from context sources. Fig. 4.3 demonstrates that with the finely dashed arrows. [87]

4.2.1.3 Federated architecture

In cases where current domains should be federated, the federated architecture is shown in Fig. 4.4 is used. For example, different departments in a city operate their NGSI-LD API infrastructure with the Context Broker, but applications should easily access all available information with just one access point. The architecture operates similarly to the distributed architecture mentioned in Sec. 4.2.1.2, unless whole domains are recorded in the respective context broker as an access point instead of simple context

sources. The domains will typically be recorded at a higher level (coarse-grained level) to the federal context registry with size scopes that can match the range provided in the requests, particularly geographic size. For example, the domain would be registered with information about entities with a building within a geographical area instead of registering individual entities, such as buildings. The applications query or subscribe to geographically-based entities, e.g., buildings in a particular city area. The domain context brokers are located within the federation server, providing information and sending the request to these brokers. They can also summarize the data, and the request gets the response as in centralized and distributed scenarios. [87] A domain itself may use a centralized or

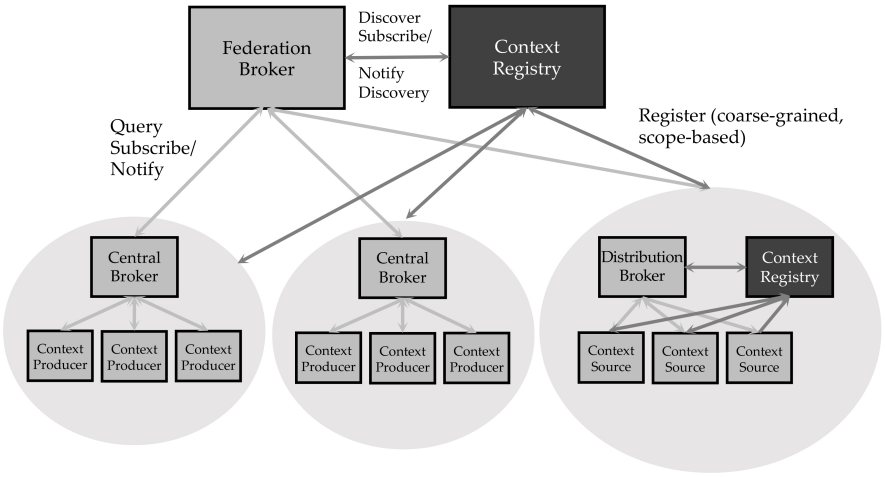

Fig. 4.4 Federated architecture [87]

distributed architecture or even a federated sub-domains architecture, as in distributed applications, which also discover relevant domains via the federal context registry and contact the context brokers directly in each domain. [87]

4.3 Fiware - Orion Context Broker

FIWARE [88] is a project sponsored by the European Commission to develop technologies for IoT and the future Internet, in collaboration with participants from the ICT sector. FIWARE's mission statement is

> To build an open, sustainable ecosystem around public, royalty-free, and implementation-driven software platform standards that will ease the development of new Smart Applications in multiple sectors [88].

The FIWARE initiative provides a set of application programming interfaces (APIs) to develop smart applications based on open and royalty-free specifications. FIWARE is based on the concepts from the previous SENSEI project and the IoT-A architecture, a European initiative to define an architecture for the Future Internet that is reusable across different domains [30]. FIWARE specifies components called Generic Enablers that constitute the building blocks for smart applications in different areas. These Generic Enablers cover aspects that include, but are not limited to, security, data management, cloud hosting, and device management.

The core of the FIWARE ecosystem is the so-called FIWARE platform. It is a set of public and free-to-use API specifications that come along with open source reference implementations. There also exists an initiative called FIWARE lab, which offers the platform in a cloud environment. Whereas the FIWARE lab is merely for testing, experimenting, and evaluation, the FIWARE iHub initiative is supposed to provide production-ready cloud services in the future. The FIWARE platform is grouped into seven major parts called the “generic enablers (GEs)” [88]. Every GE represents a particular aspect of FIWARE services and provides one or more components and reference implementations that support the specified APIs. Additionally, there are so-called “domain-specific enablers (DSEs) that (will) provide components for specific domains like health, energy and Industry 4.0.

The *Orion Context broker* allows the user to manage the entire life-cycle of context information, including updates, queries, registrations, and subscriptions [88], which are also used in DezCom with transactions on assets approach. The FIWARE Catalogue contains a rich library of components (Generic Enablers) with reference implementations that allow developers to put into effect functionality such as connecting to IoT or Big Data analysis, making programming much easier [88].

4.3.1 Features

The general enablers are organized as follows [89]:

- Data/Context Management: This contains all components that are needed to store, access, process, and analyze data as part of a smart application
- IoT Services Enablement: Here are all components needed to set up sensor networks and route sensor data to other GEs.

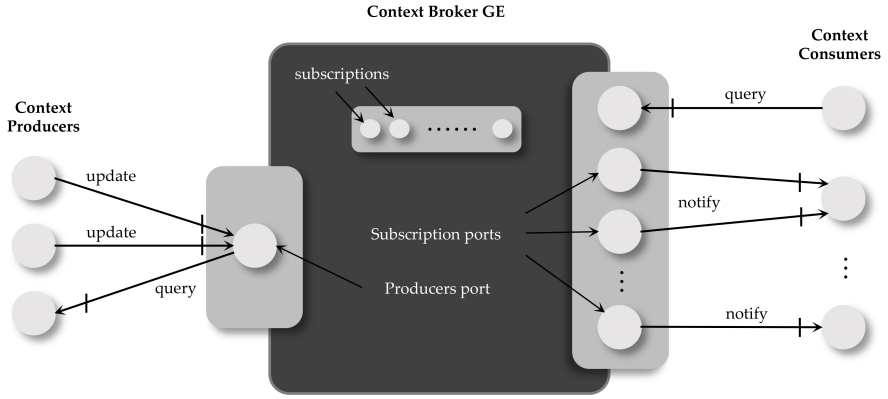

Fig. 4.5 Context brokers communication flow Fiware [88]

- Advanced Web-based User Interface: Components to design user interfaces, including geographical information and interactive 3D charts
- Security: Components to add, define and enforce declarative security
- Advanced middleware and interfaces to Network and Devices
- Applications/Services and Data Delivery: Components and tools for data visualization, easy generation of mashups, and app-store-like distribution of services and data
- Cloud Hosting: Components and tools aiming at providing and managing FIWARE services via a cloud infrastructure

4.3.2 Communication Paradigm

The communication paradigm is centralized and runs on top of a TCP/IP network. The Orion context broker is deployed in a cloud environment, as mentioned in Deployment 4.3.3, where issues of centralization can be mitigated using a distributed architecture. Even though parts of the stack are deployed reliably in a distributed manner, there is still the case for a single point of failure when the network connection is lost or when the cloud environment is not reachable by the IoT devices. The communication primitives are not limited to the REST API interface as the Orion context broker leverages the concept of generic enablers.

4.3.3 Deployment

FIWARE is an open-source software consortium that develops and curates software. As mentioned before, one such software is the context broker called the Orion Context Broker. It is developed for massively scalable applications such as smart city applications. Therefore, the context broker must be deployed in a highly available server infrastructure. Cloud infrastructure creates the environment required for such highly scalable applications where the software can be scaled as the application grows in the real world. The Orion context broker's software stack has a REST API interface developed as per the NGSI standard and is exposed using a reverse proxy through a web-server. The software stack orchestration is relatively simple, but there are multiple dependencies for each layer of the stack. A local database is required as a persistence layer for the context broker. In many applications, it is initially suggested to deploy the database in the same server as the REST API server. As and when scaling is necessary for the application as it grows in use and memory, an horizontal scaling architecture with distributed deployment is advised where each component of the stack has its resource and can be scaled for each of the necessary resources with a protected cloud environment. The most necessary cloud server resources are CPU for computing and execution, bandwidth for network transfer, and memory where both the storage and RAM are considered here. The database grows over time, and it is advised to run on servers where the resources for memory can be scaled. The REST API is the interface to handle the bandwidth for network transfer and concurrent connections from devices deployed in the world. For such an application, the REST API interface is deployed with a load balancer to distribute the incoming connections within deployed application servers.

4.3.4 Flexibility

The flexibility of using the Orion context broker is achieved using the concept of Generic Enablers (GE). It extends the FIWARE core components, where the GEs are pluggable to the primary system for interfacing, information processing, or scheduling tasks. The concept of Generic Enablers increases the deployed software's complexity but improves interface options with other systems.

FIWARE used a great variety of different programming languages (C++, Java, Python, NodeJS, ...) and environments for developing their reference implementations. Fortunately, the FIWARE community provides docker

images for every component, which makes dealing with different runtime requirements relatively easy [90].

4.3.5 Limitations

FIWARE's Orion Context Broker is a software stack with multiple interfaces within the software stack, as in distributed micro-services architecture. As the system scales, it has to be deployed in a distributed manner for efficient scaling and optimized use of resources. The stack has a REST API interface that interacts with the clients connecting with the context broker. It also has a MongoDB NoSQL database deployed in the same server that needs to be deployed in a distributed manner when the system is scaled. Even though FIWARE is the only available multi-purpose context broker, the architecture is centralized, as shown in Fig. 4.5 with a single point of failure. With every interfaced entity having a web-server listening on a port for REST endpoint calls for every communicated message. QoS provided by the Context Broker improves when using the sharded MongoDB cluster compared to the scale-up configurations of MongoDB [90].. Issues with scaling, effort to scale the systems in a distributed manner, and the amount of resources required to manage and run the system due to the complexity it provides are considered as limitations for using the system.

4.3.6 Performance

The performance of FIWARE's Orion context broker for huge-scale projects is limited because scaling becomes an issue. Reactive scaling of the deployed servers is the only way to scale the Fiware system for any surge in demand. In the literature about the performance evaluation of FIWARE [90], various experiments were performed. Few of the results that are listed as requirements for the development of DezCom are the following:

- Effect of using increasingly larger instances for Orion Context Broker, keeping the database instance (i3.xlarge) fixed. Performance with the largest evaluated instances for the Context Broker and MongoDB is only 30% better than the baseline configuration, despite having 4x more CPU and memory (and costing almost 4 times as much) [90].
- FIWARE's MQTT IoT Agent component also crashes beyond 1000 requests per second [90].
- FIWARE's agent crashes and is unstable after the system is overloaded [90].

- The memory used by the IoT Agents spikes before they crash, when the system is overloaded [90].

4.4 MQTT

MQTT is an ISO standard (ISO/IEC PRF 20922) messaging protocol [91]. Andy Stanford-Clark of IBM and Arlen Nipper of Cirrus Link co-authored the first deployed version of MQTT in 1999 [91]. Since then, it has been used in various remote sensing and telemetry projects. If not whole, then a subset of the messaging techniques was abstracted into applications for implementation. In 2013, IBM submitted MQTT v3.1 to the OASIS specification body with a charter that ensured that only minor changes to the specification could be accepted [91]. The MQTT standard has a publish-subscribe-based messaging model for communication. The topics are used to communicate between the nodes in a network. The flow of communication is highly event-based, where a broker is always listening to topics that are published and posts the messages to the subscribed clients. The messaging protocol uses existing TCP/IP standards, making it easier to deploy these in existing TCP/IP based networks [91]. It is designed for connections with remote locations where a ”small code footprint” is required, or the network bandwidth is limited [91]. Most of the implementations related to this work were carried out using the python library, which was implemented using the MQTT specification. The publish-subscribe model of the MQTT requires a broker, which federates the network receiving all the messages and relaying them to all the subscribed clients. It also manages all the nodes’ connection and status as a publisher or a subscriber for every topic. There is a variation of the MQTT protocol available for low-power, low bandwidth devices called the MQTT-SN. It was aimed at non-TCP/IP based networks where MQTT could still be used. There are deployable implementations of MQTT-SN in ZigBee and ContikiOS for embedded devices. When a publishing client first connects to the broker, it can set up a default message to send to subscribers if the broker detects that the publishing client has unexpectedly disconnected from the broker.

4.4.1 Features

The messaging protocol can be used on resource-constrained devices where the messaging bandwidth is less (<1Mbps) or requires only a small code

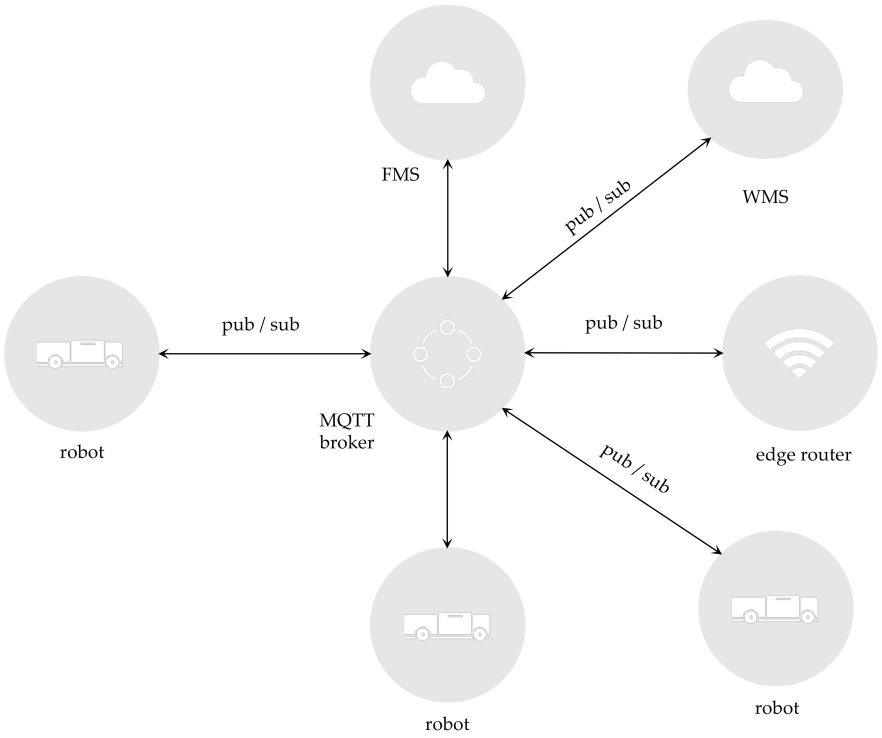

Fig. 4.6 MQTT stack as deployed at the research facility of the Chair for Material Handling and Warehousing [56]

base for communication and messaging [91]. A minimal MQTT control message can be as little as two bytes of data [91]. A control message can carry nearly 256 megabytes of data if needed [91]. There are fourteen defined message types used to connect and disconnect a client from a broker, publish data, acknowledge the receipt of data, and supervise the connection between client and server [91]. MQTT offers a generic communication interface and transports the messages to subscribers with a central MQTT broker's help. The following are the three important communication primitives required to successfully communicate with a MQTT broker, as illustrated in fig. 4.7.

Connect Example of an MQTT connection (QoS 0) with connect, publish/subscribe, and disconnect. The first message from client B is stored due to the retain flag. The client waits for a connection to be established with the server and creates a link between the nodes.

Disconnect The client waits for the MQTT client to finish any work it must do and the TCP/IP session to disconnect.

Publish Returns immediately to the application thread after passing the request to the MQTT client.

4.4.2 Deployment

The MQTT broker is deployed in a central service. The most used MQTT broker is the Eclipse Mosquitto broker. At the research facility at FLW, TU Dortmund, the Eclipse Mosquitto broker is deployed on a docker for communicating with different entities in the industrial network. Eclipse Mosquitto is an open-source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1, and 3.1. Mosquitto is lightweight and suitable for all devices, from low-power single-board computers to full servers. MQTT can be installed easily in Linux servers with a simple configuration and accessed using a web-socket port.

4.4.3 Communication Paradigm

As shown in Fig. 4.6, a deployed MQTT stack consists of two main parts, namely, the nodes and a broker. There is no difference between a subscribed node and a publisher node. Nodes communicate with a server, often called a broker. When a node needs to connect to another node, the nodes must be connected to the broker since the broker controls the messaging. There is no peer-to-peer networking. Therefore, every message published in a topic will be sent to the broker and the broker will then send the message to the nodes that are subscribed to the topic.

Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the connected broker's data, as illustrated in Fig. 4.7. The broker then distributes the information to any client that has subscribed to that topic. The publisher does not need to have any data on the number or location of subscribers, and subscribers, in turn, do not have to be configured with any data about the publishers.

If a broker receives a topic for which there are no current subscribers, it will discard the topic unless the publisher indicates that it is retained. This allows new subscribers to a topic to receive the most current value rather than waiting for the next update from a publisher.

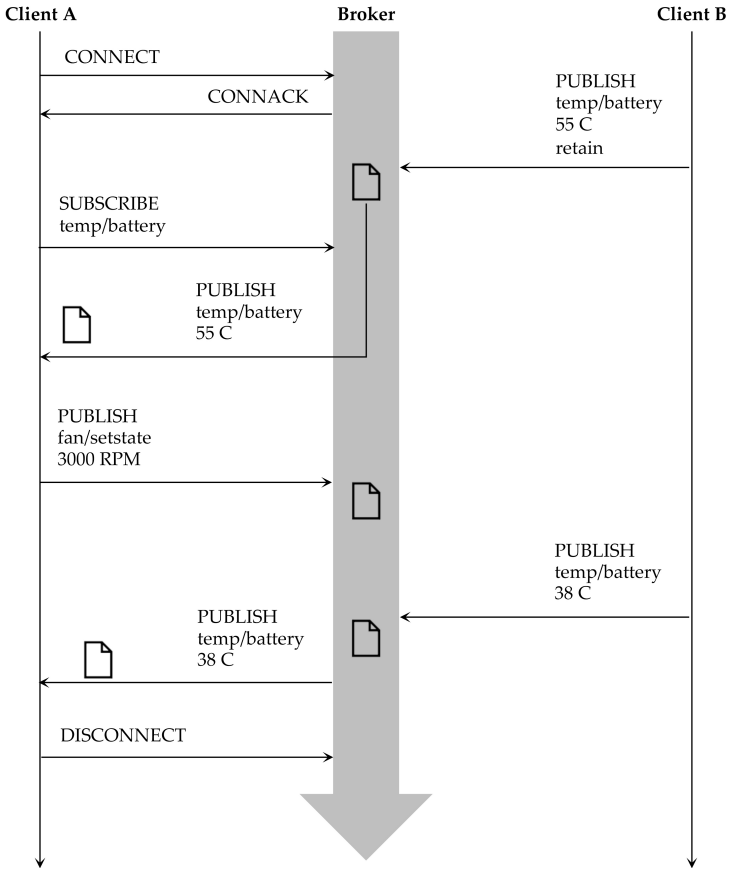

Fig. 4.7 Message flow in a MQTT broker

4.4.4 Flexibility

The MQTT communication platform strives to be a simple publish-subscribe communication system that is centralized. It requires a single computer that is reachable by the rest of the clients over a TCP/IP network. It is effortless to set up and requires very few code lines to set up the communication. Since it acts as a communication library and not as a framework, the developer will create a contextual framework for the broker depending on the application.

4.4.5 Limitations

Clients only interact with a broker, but a system may contain several broker servers that exchange data based on their current subscribers' topics. MQTT relies on the TCP protocol for data transmission. A variant, MQTT-SN, is used over other transports such as UDP or Bluetooth. MQTT sends connection credentials in plain text format and does not include any measures for security or authentication. This can be provided by the underlying TCP transport using measures to protect the integrity of transferred information from interception or duplication. MQTT is primarily a messaging system where context information can be embedded in the messages. One of the design limitations in deploying MQTT brokers is that the number of socket connections depends on the number of files that a process can open concurrently. This is limited in Linux by default to 1024. Since the Mosquitto broker is a single-threaded process, for networks with more than 1000 publisher clients with a payload rate of 10 messages per second, the brokers' functionality is reduced due to the network's limitations the operating system.

4.4.6 Performance

As the MQTT broker is a single-threaded application, the limitation of network input and output is limited with the number of concurrent connections. When there is a significant amount of concurrent connections in the order of 1000 and more, the process has minimal time to serve all the publishers and subscribers. As the number of connected devices increases, the number of concurrent connections increases the amount of network input-output operations also includes disk input-output adding to the overhead of the MQTT broker servicing connections faster. The file limitations and the single-threaded execution limits scaling at a determined upper limit for a MQTT broker deployment. The ability to predict the delay based on the number of publishers (e.g., sensor data sources) is critical to estimate an IoT system's overall performance and the trade-off between speed and accuracy [92].

4.5 What is DezCom?

In this section, a solution is proposed for decentralized communication of entities in an Industry 4.0 environment. Entities are heterogeneous actors

which can be considered as material handling systems in this context. In the field, these actors need to perform tasks collaboratively within a warehouse. For accomplishing the tasks in a warehouse, these systems need to discover and associate with the network. Once they are part of the network, the robots, sensors, and other material handling machines can receive messages from coordinators of the warehouse, such as a warehouse management system, and can perform various material handling tasks. The network is available for the transfer of payload between the systems. When there is a single entity with the same specification of features and the tasks it can perform, it becomes easier for the rest of the systems to understand and act in an industrial network. Since this is not the case, and there are multiple robots and a multitude of systems involved in a material handling system, the curation of the messages, tasks, and the distribution of tasks becomes rather complicated. A context manager must perform the protocol specification of discovery, network association, and advertisement. This context manager handles all the messages in a secure manner and forwards it to the system software for further processing. In the case of robots, the messages are received, and the robot's core system software understands the delivered messages and performs the tasks as instructed by the received messages. For this purpose, context management is required to abstract the communication between the systems. This helps in the self-organization of the systems within the warehouse for various tasks. Planning and scheduling these systems are performed using resources in the network where they run as a software instance. For these software instances to understand the current status of the network i.e. the various advertised properties of the entities with their availability, a software networking paradigm is necessary to transport these various messages between heterogeneous systems. This will enable the management of complex systems where heterogeneous systems take part in performing tasks.

4.5.1 Requirements for a context broker development

The requirements for a decentralized context broker must include but must not be limited to the following:

High throughput message fan out - a small number of data producers (publishers) need to frequently send data to a much larger group of consumers (subscribers).

Addressing, discovery - sending data to specific application instances, devices, or users.

Load balancing with N-way scalability where the application(s) produces a large volume of work items or requests and dynamically uses a scalable pool of workers.

Location transparency - applications need to scale to a very high number of instances spread out geographically, and with intrinsic modularity in the applications, specific endpoint-configuration information for applications to understand the endpoints.

Fault tolerance and Trust architecture - the application needs to be highly resilient to network or other outages with the application executed by diverse entities of the process supply chain to allow anyone to participate in the communication.

Decentralized consensus is considered as a by-product of such a highly scalable distributed fault-tolerant architecture. The above two widely used systems do not meet the requirements except for high throughput message fan-out, i.e., if the server with the central instance is unreachable, the whole network communication is disrupted.

4.5.2 Motivation for a decentralized context broker

In the current state of the art, there are many applications in the industry that require a service, which in most cases, is a database where applications query and store information. Most of the databases available are centralized databases that have a central entity. Even in federated databases, geographical decentralization takes place but not for the database availability itself. An abstracted view of central databases can be illustrated as in Fig. 4.8 with an input source and an output. The server or the database is protected using a firewall, which helps the service be protected from external threats. However, it is still vulnerable to other kinds of scaling issues and network breaches. Such an architecture is considered vulnerable. In these cases, when the communication layer is made secure, and all the data transferred by the clients are authenticated and logged, it becomes easier to develop scalable systems with latent features for network security.

In this architecture, every client has a string that can be considered as a password. These passwords act as authentication, which makes firewall breaches easier when the password is known. The service, which could be a database or compute node, could prove to be vulnerable.

Therefore, a proposal for a decentralized architecture for systems communication is proposed, as illustrated in Fig. 4.9. Since the inception and the proof of ledger-based systems, it has become inevitable to use blockchain technology to decentralize systems communication. The architecture proposes that all the communication is wrapped using a

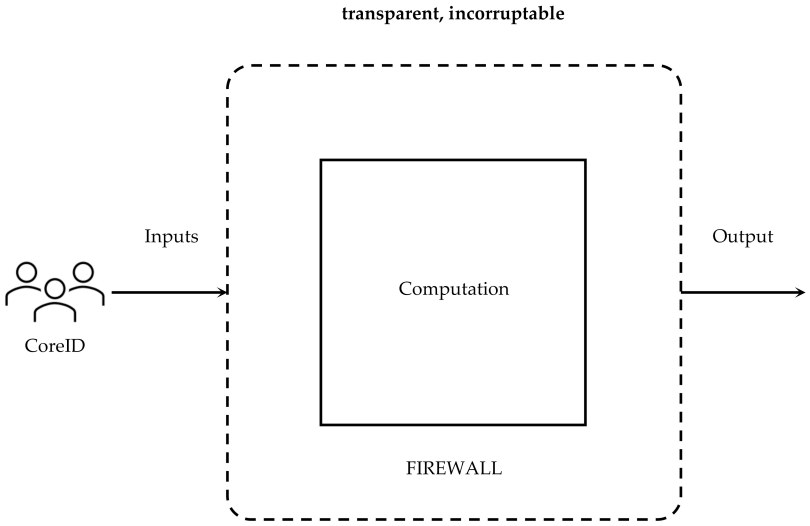

Fig. 4.8 Vulnerable architecture with a centralized architecture [88]

distributed ledger technology that provides the required security layer for communication.

An in-detail deep dive into distributed ledger technology, blockchain followed with various consensus models are presented in this chapter.

In Fig. 4.10, a use case for having a decentralized context broker is specified where the main objective is not to share the data but only the answers. There are multiple algorithms where computations and results can be arrived at without having access to all of the data. Data is an essential commodity when autonomous machines have to perform intelligent actions and make them economically viable. Even though data privacy is one of the motivational factors for developing algorithms that can compute answers without access to data, the economic value involved in controlling data access is another major factor for developing such decentralized communication technology. This involves the owner of the data to become a stakeholder in making optimized decisions during industrial operations. In a naive sense, it can be seen as monetizing the collected data during industrial operations.

An example of an algorithm is laid out to clearly understand the requirement of a decentralized context broker and its importance in an industrial scenario. We consider that three entities are continuously collecting data independently of each other. The three entities are autonomous machines from three different manufacturers. They are not intrinsically sharing the data, but they are working in parallel for the same customer who has deployed these machines in the industrial facility. In an

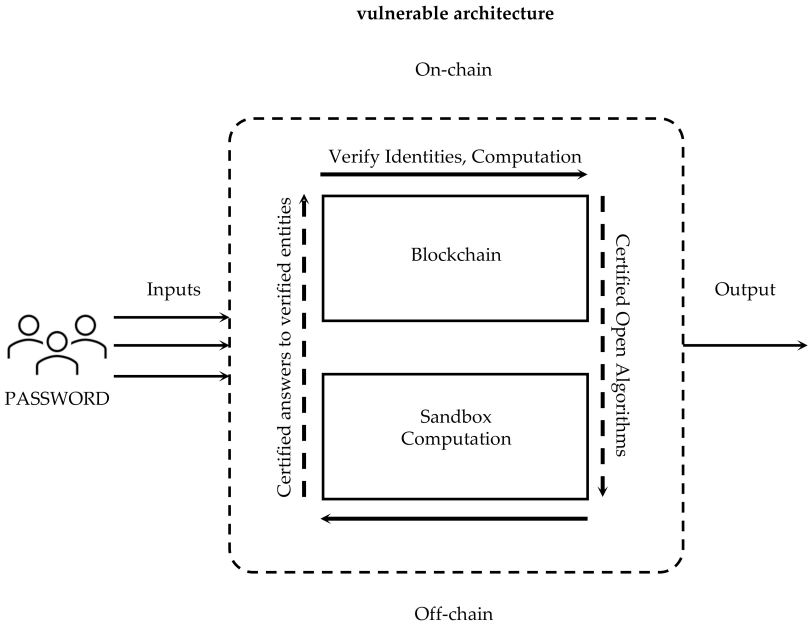

Fig. 4.9 Secure decentralized architecture using blockchain [88]

initial sense, the data would be transported to a central server where the data is stored, consumed and acted upon by the customer. Suppose the data needs to be shared between the operating machines. In that case, it is only possible if the customer requires the manufacturers to work amongst them to implement a data sharing policy. Unfortunately, the data is not accounted for since the manufacturer of the machine fulfills the contract by providing reliable operations in the facility. This data improves the operations and the underlying processes, but the machine does not have any accounting mechanism to have access to the data. The following multi-party computation developed to share data between machines illustrates a transparent data sharing policy scenario and the requirement for a context broker that runs on blockchain technology to account for the data exchanges. Consider that each machine in the field has a number, which is a KPI. This KPI is also available at each of the three machines. When an average is calculated between this KPI, there is potential for optimizing the machine's operations and providing economic value for the customer. As per our assumptions for decentralizing communication between autonomous machines, the data is not shared but only the answers. For this purpose, machine 1 adds a random number to the KPI. So does each of the machines with the KPI. In our scenario, there are three machines; therefore, there are three pseudo KPI. Let us assume that x_1, x_2 and x_3 are the three

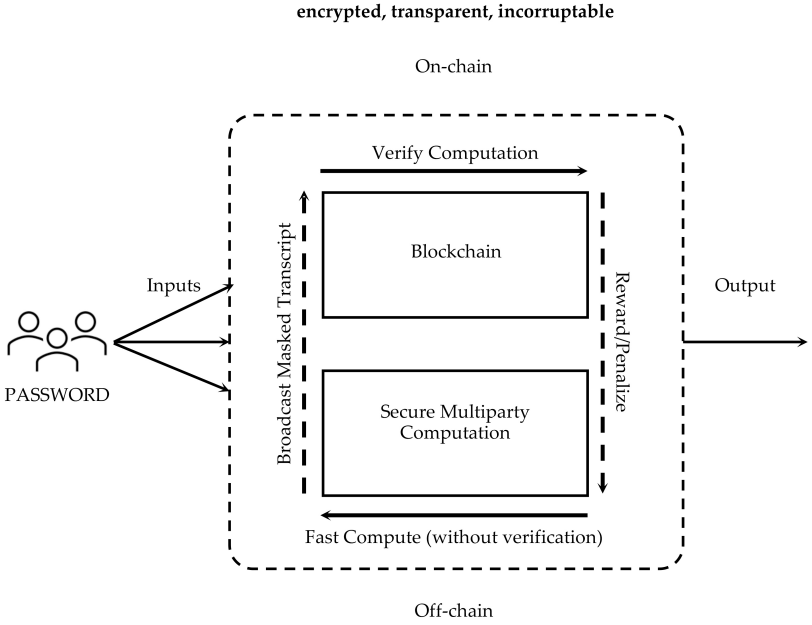

Fig. 4.10 Architecture for secure multiparty computation using blockchain [88]

KPI and y_1, y_2 and y_3 are the three pseudo KPI. This data is shared amongst the machines. Since there is a random number added to the actual KPI, the value is intrinsically flawed. However, to compute the average of that number, machine 1 sends y_1 to machine 2. Machine 2 adds y_1 to y_2 and so does machine 3. Now since all the numbers summed together are not the right sum, another communication round is made where each machine subtracts their random number and broadcasts. When each of the machines does this, they all have the sum of the numbers without actually sharing the KPI. Now, by dividing the sum of KPI with the number of participating machines, we can quickly arrive at the average. The average is known to all the machines but not the individual KPI, which means the answer is arrived at without sharing the data itself. This example was presented by Dr. Alex "Sandy" Petland, professor at MIT Media Lab for Ethics and Human dynamics. This algorithm is Turing complete with which any computation can be performed without sharing the data. The critical underlying mechanism is to provide a reliable communication layer where each of the communication rounds is logged with metadata from each machine to understand the computation's current state. The computation has multiple steps to be performed and multiple broadcast rounds of sending a payload. With the help of blockchain, it becomes easily trackable, reliable to transport data. With the features of a context broker, it becomes easier to transfer the

messages between specific sets of nodes and store them reliably. A context broker is a communication software that provides a robust architecture for reliably transporting payload between the systems.

There are various aspects involved in developing such a software that can handle communication from heterogeneous entities with a wide range of communication capabilities. For understanding the fundamentals of such communication software, initially, two types of such communication software are studied in Sec. 4.1. There are two predominantly used industrial systems communication software called context broker. They help to scale the number of devices connected to a system both horizontally, i.e., heterogeneous device types that connect to the network, and vertically, i.e., scaling the number of similar devices that belong to the network. The two types of systems are MQTT and Fiware which are described in Sec. 4.4, 4.3 respectively. Followed by context brokers, different types of systems in communication and network topology must be understood, described in the following Sec. 4.2. With this understanding of different systems, the various state of the art system software are analyzed in Sec. 4.8.1 to understand the requirements for developing a highly scalable, heterogeneous communication platform. Followed by technology status evaluation, choice of a software stack, development, implementation, and finally, performance analysis.

4.6 Inevitability of Blockchain

A reliable networked computer system with distributed nodes must cope with the failure of one or more of its networked nodes. A failed node may exhibit a type of behavior often overlooked, i.e., sending conflicting information to different system parts. Sending conflicting information could be delayed information relevant to the system's operations but triggers an erroneous action due to transmission delays, a corrupt message, or a malicious node in the network. The problem of coping with this type of failure is expressed abstractly in the Byzantine Generals Problem.

4.6.1 Byzantine Generals Problem

The Byzantine Generals Problem is a term etched from the computer science description of a situation where involved parties must agree on a single strategy to avoid complete failure, but where some of the involved parties are corrupt and disseminating false information or are otherwise

unreliable [93]. The Byzantine Generals Problem makes for an excellent fundamental example of blockchain-based consensus algorithms. Understanding it generally elevates the comprehension of other consensus algorithms. The major problem of distributed and decentralized systems is that a network system is disseminating false information. The Byzantine Generals Problem can be explained with the following scenario explained by Leslie Lamport [93]. Several divisions of the Byzantine army are camped outside an enemy city. A general commands each division that is camped outside the enemy city. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a joint plan of action. The dilemma assumes that each general has its army and that each group is situated in different locations around the city they intend to attack. The generals need to agree on either attacking or retreating. It does not matter whether they attack or retreat, as long as all generals reach consensus, i.e., agree on a common decision to execute it in coordination. The common plan of action may be one of the two: attack or retreat. If they do not attack simultaneously around the city, the attack will be futile. Here the messenger plays a vital role in delivering the message on time with the decision intact. Only then, the generals will be able to converge to a consensus in the decision. There might also be traitors amongst the generals. Therefore all loyal generals should reach a consensus.

A simplified version of this consensus problem can also be presented in the following [93]: Each general has to decide: attack or retreat (yes or no); After the decision is made, it cannot be changed; All generals have to agree on the same decision and execute it in a synchronized manner. As mentioned earlier, the communication problems are related to the fact that one general can only communicate with another through messages, which are forwarded by a courier. Consequently, the Byzantine Generals Problem's central challenge is that the messages can get somehow delayed, destroyed, or lost. If we apply the dilemma to the context of blockchains, each general represents a network node, and the nodes need to reach consensus on the current state of the system. In another perspective, most participants within a distributed network have to agree and execute the same action to avoid complete failure. Besides, even if a message is successfully delivered, one or more generals may choose (for whatever reason) to act maliciously and send a fraudulent message to confuse the other generals, leading to a total failure. Therefore, the only way to achieve consensus in these types of a distributed system is by having at least two thirds or more reliable and honest network nodes. This means that if most of the network decides to act maliciously, the system is susceptible to failures and attacks (such as the 51% attack).

In a few words, Byzantine fault tolerance (BFT) is the property of a network or a communicating system that can resist the class of failures derived from the Byzantine Generals Problem. This means that a BFT

system can continue operating even if some nodes fail or act maliciously. There is more than one possible solution to the Byzantine Generals Problem and, therefore, multiple ways of building a BFT system. The Byzantine Generals Problem is an intriguing dilemma that eventually gave rise to the BFT systems, which are being extensively applied in various scenarios. Beyond the blockchain industry, a few use BFT systems cases include the aviation, space, and nuclear power industries. In blockchain technology, there are different approaches for a blockchain to achieve Byzantine fault tolerance, leading us to the so-called consensus algorithms. In this work, blockchain technology is used due to the intrinsic nature of arriving to a consensus amongst the decentralized nodes. The other features that blockchain technology provides make it an inevitable choice for developing a decentralized context broker.

4.7 Problem Application Fit

This section will explore and evaluate if blockchain is the right application for the developed context broker. The context broker has requirements derived from existing centralized context brokers. We will use those requirements presented in Sec. 4.5.1 and evaluate them qualitatively to understand if blockchain caters to the requirements for developing a decentralized context broker. Since blockchain technology is still recent, several organizations are investigating ways to integrate it into their company [94]. The fear of missing out on this technology is powerful, and most companies are treating the issue as ”we want to use blockchain somewhere, where can we do it?” [94], which leads to frustration with technology because it cannot be universally applied. A better approach would be to understand blockchain technology first, where it fits, and then identify (new and old) systems that might fit the blockchain paradigm [94].

Blockchain is not limited to the domain of cryptocurrency, but it is also not a single solution that can solve all the application problems. We take the guidelines provided by Yuga et al. [94] for understanding the compatibility of blockchain for applications that are not cryptocurrency applications. Blockchain technology solutions may be suitable if the activities or systems require features. It will be quantitatively analyzed if the application that we develop, i.e., the DezCom context brokers, requires a subset of these features. The features named by Yuga et al. are written in italics, followed by a simple explanation if necessary, and finally, the analysis of the application-specific understanding is provided.

Many participants: an application that requires many participants irrespective of their nature of how they are deployed in an industrial

application. Moreover, the participants can scale from a few to hundreds of clients as in a context broker case.

Distributed participants. The participants are distributed geographically as in physical entities and distributed in terms of information technology. There are multiple agents deployed, even in a single computer that interfaces using a context broker system.

Want or need for lack of trusted third party. As mentioned earlier, when many participants are part of a network, it becomes hard to keep track of the parties associated and disassociating from and out of a system, respectively. Additionally, not all participants are part of the network that will allow the network to function continuously. Therefore, there is an inherent lack of trust between the functioning parties in a context broker.

Workflow is transactional (e.g., transfer of digital assets/information between parties). In general, Industry 4.0 creates a data economy, where the data generated by the systems is a valuable asset. Moreover, the generated data needs to be reliably transmitted between the participating clients for actionable tasks within an industrial system. Therefore, the data in a context broker can also be considered transactional.

A need for a globally scarce digital identifier (i.e., digital art, digital land, digital property). The data produced requires a chronologically sorted unique digital identifier, which will be used to identify the messages and distribute them when systems are participating in the network requests. A well functioning context broker allows for reliably transmitting information across the network and delivering data on demand. For such purposes, the data, which is the asset, will require a digital identifier.

A need for a decentralized naming service or ordered registry. Since heterogeneous industrial systems are interacting within an industrial network, service discovery and peer to peer communication using a namespace is necessary to reduce the overhead on the participating systems to store such information before joining the network.

A need for a cryptographically secure system of ownership. A context broker would not critically require a security system that can be cryptographically proven. Still, to improve transparency and resolve conflicts when there are malfunctioning systems, it is necessary to determine the ownership of the produced data with cryptography. Furthermore, the data economy's importance can be realized using such a context broker when the accountability for generating the data can also be kept as a record.

A need to reduce or eliminate manual efforts of reconciliation and dispute resolutions. In a highly scalable context broker, data dispute resolution is necessary and needs to be performed automatically. Adding to the complexity of scalability, when the systems are highly mobile and distributed, blockchain applications provide robust application features for automatic data reconciliation.

A need to enable real-time monitoring of activity between regulators and regulated entities. When heterogeneous machines communicate with multiple vendors, the context broker should allow features for tracking down malicious agents and bad actors in the network.

A need for full provenance of digital assets and full transactional history to be shared amongst participants. When the system enables the creation of various digital assets, and the assets are being transacted between machines for executing tasks in an industrial scenario, full transactional history should be accessible amongst the participating nodes.

Permissioned blockchain networks may disclose blockchain data publicly. Only those inside the blockchain network will have the data available. Consider cases where data could be regulated by legislation or regulations (such as Personally Identifiable Information (PII) or the Regulation on General Data Protection (GDPR)). Even within a permissioned blockchain network, data such as this may not be suitable to be stored [94].

False Data Input – Because multiple users connect to a blockchain, some can send incorrect data, imitating data from legitimate sources (such as data from sensors). Data verification, which enters a blockchain network, is difficult to automate. Implementation of smart contracts may provide additional checks to assist in validating data where possible [94].

Node Diversity – A blockchain network is only as strong as the sum of all of the current nodes participating in the system. If all nodes share similar hardware, software, geographic location, and a messaging scheme, then there is some risk associated with the possibility of undiscovered vulnerabilities in security. This risk is mitigated by decentralizing the heterogeneous device network, which could be defined as "the non-shared characteristics between any single node and the generalized set." [94].

The features, as mentioned above, are part of any decentralized blockchain system. Still, the features of the blockchain, as discussed by Yuga et al. [94], are not limited to the points mentioned above. However, we can already conclude that blockchain is a candidate for developing decentralized consensus for a context broker. In the following sections, blockchain's fundamentals are discussed along with various candidates for implementing decentralized consensus among the nodes. It is the primary building block of a decentralized context broker.

4.8 Blockchain Technology

A blockchain is a particular type of decentralized database. There are many other definitions from the realm of cryptocurrency and other industries. However, the core idea and the perspective we look at blockchain in this

research work are blockchains as a decentralized database for storing digital assets. A blockchain is commonly operated with decentralized, networked nodes because the database gets clunky, and there are superior alternatives for centralized and managed database solutions. Blockchain technology's real potential can be exploited in a decentralized environment – that is, one where all users are equal, as seen in a socially networked industry, as developed under the Industry 4.0 framework for providing autonomy to multi-vendor heterogeneous collaborative systems. The blockchain can't be deleted or maliciously be taken over due to the nature of connected blocks where the database always refers to the previously stored information. All the nodes are in synchrony with the replicated states, and as discussed earlier in Sec. 4.6, the nodes communicating in this network are BFT secure. The blockchain database becomes a single source of truth that anyone can see in an industrial network. A blockchain has specific unique properties. There are sets of rules for committing data into the network or the database. Data is added over time in structures called blocks. Each block is built on top of the last and includes information that links back to the previous one. By looking at the most up-to-date block, we can check that it has been created after the last. So if we continue down the "chain," we'll reach our very first block – known as the genesis block.

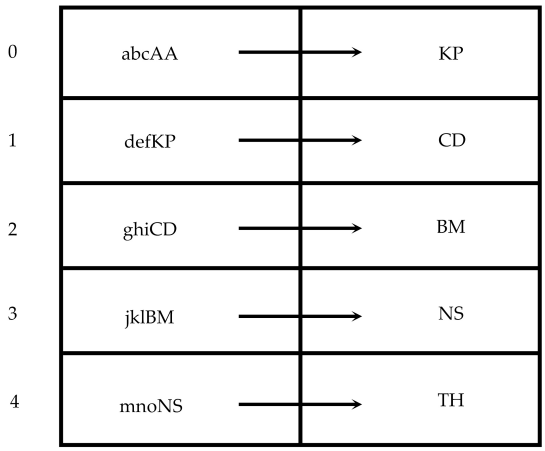

Fig. 4.11 A example database using excel sheet where each entry is linked to the last.

To understand the linked nature of a blockchain database, let us consider a spreadsheet with two columns. In the first cell of the first row, the stored data is Robot-A. In this example, we create a rule that an identifier is created using a function generated concerning the data. Using the function along with the first cell's data, a two-letter identifier is generated, which will then

be used as part of the next input. In this example, the two-letter identifier KP was generated concerning the first cells stored data. This identifier must be used to store the next cell in the second row (Robot-BKP). This means that if anyone changes the first input data (Robot-C), the function will generate a different combination of letters in every other cell. Looking at row 4 now, our most recent identifier is TH. As was mentioned earlier, it is not possible to go back and remove or delete entries? That is because it would be easy for any participating node in the network to identify that the data has tampered with. Such an attempt would not only be identified, but also data would be ignored for storage. Certain blockchain systems keep count of how many times a particular node attempts to tamper with the data and controls how the system interacts with the blockchain to ensure the network's reliable operation.

Suppose anyone changes the data in the very first cell. The same data will get a different identifier in the next request, which would mean the second block would have different data, leading to a different identifier in row 2, and so on. Once the data has been stored, it is virtually impossible to modify or delete it. One way of deleting already stored data is to deem the data invalid inside the data structure; thus, the history of communication is maintained without any fraudulent changes in centralized database systems. In centralized, managed database systems, the database administrator can log in to the database and change the previously stored values and manipulate the system without any trace, which is a technical possibility. Whereas in blockchain, even this manipulation is stored/logged within the blockchain database.

4.8.1 Blockchain Categories

It is possible to categorize the networks of blockchain based on their authorization model that decides who can hold them (e.g. publish blocks). It's *Permissionless* if anybody can publish a block to the blockchain. When blocks can only be published by users that are authenticated, such types of blockchains are called *Permissioned*. A Permissioned blockchain network is more like a regulated company intranet, while a Permissionless blockchain network is like the public internet, with everyone being able to participate. Simply defined as, Permissioned blockchain networks are frequently used for a consortium of organizations and individuals. The distinction between the kinds of blockchain systems is important since it has an influence on the components of the blockchain.

4.8.1.1 Permissionless

Blockchain networks with no permissions are decentralized blockchain platforms open to anyone who publishes blocks without authorization. Most open-source software is open access, free of charge to anyone wishing to use the permissionless blockchain platforms. Since everyone has the right to post blocks, it results in the property that everyone can read the blockchain and issue transactions on the blockchain. Any blockchain network user can read and write to the directory inside a blockchain network without permission. Since unauthorized blockchain networks are open to anyone, malicious users can try to publish blocks in such a way to subvert the system (Sybil attack[1]). To avoid this, blockchain networks without authorization sometimes use a multi-party agreement or consensus, demanding that users use or retain resources when publishing blocks. The consensus is detailed in Sec. 4.8.2 as it is the fundamental building block for a permissionless blockchain. It ensures that malicious users can easily compromise the program. Proof-of-work is elaborated in Sec. 4.8.2.1 and Proof-of-stake (see Sec. 4.8.2.2), these are the consensus models which are widely implemented for any blockchain. Consensus systems in blockchain networks without permission typically support non-malicious actors by rewarding protocol block publishers with a native cryptocurrency.

4.8.1.2 Permissioned

The Permissioned blockchain networks must be approved (whether centralized or decentralized) by users or a group that runs the blockchain to ascertain who publishes blocks. Since blockchain is kept only by Permissioned users, access to reading the blockchain can be restricted, and the transaction history for users outside the permissioned realm. Registered blockchain networks can either allow anyone to read the blockchain or restrict read access to allowing it only to the approved users. It also permits anyone to transact to the blockchain or restricts access only to approved persons.

Permissioned blockchain networks may have the same traceability of digital assets as they pass through the blockchain. Additionally, they are distributed, resilient, and redundant with their data storage system as in Permissioned blockchain networks. Consensus models are also used in Permissioned blockchain systems for publishing blocks. However, these

[1] A Sybil attack is an online security breach where a hacker utilizes multiple accounts, nodes, or systems to take over particular network. Blockchain Sybil attacks of cryptocurrencies are carried out by running numerous nodes on a network to achieve a majority, at least 51%, control over the network.

methods often do not require resources to be spent or maintained (as is the case with current Permissionless blockchain networks). This is because establishing one's identity is required to participate as a member of the Permissioned blockchain network; those maintaining the blockchain have a trust level since they are all Permissioned to publish blocks and since their authorization can be revoked if they misbehave. Consensus models in Permissioned blockchain networks are then usually faster and computationally less expensive.

Permissioned blockchain networks may often be used by organizations that wish to track and secure their blockchain due to data privacy and the importance of the data containing confidential business information. However, an organization decides who may publish blocks. The network consumers would need to maintain faith in that agency as the blockchain's privacy is breached as soon as an entity that has access to the blockchain is breached. Permissioned blockchain systems can often be used by organizations that want to operate together but cannot completely trust each other. Even though trust can be enforced with contracts, human error, and the ability to scale from tens to thousands of transactions per second paves the way for a mutually agreed distributed ledger. Business organizations and consortia will decide on the agreement format to be used, depending on how much overhead is required. This is to enforce trust between the interacting organizations instead of the complexity and cost. Beyond confidence, approved blockchain networks offer clarity and analysis to advise business decisions further and keep malicious agents and parties held accountable. This may explicitly include auditing and supervisory bodies that make audits a regular event rather than a periodic event.

Any approved blockchain network endorses the right to selectively disclose transaction details depending on the blockchain network users' identification. Typically, the identification is performed using asymmetric cryptographic keys proven reliable and hard to compromise, given the current status-quo of computational technologies. With this feature, some degree of privacy may be gained in transactions. For example, it may be that the blockchain documents that a transaction has taken place between two blockchain network participants, but the specific substance of the transactions is only available to the parties concerned.

Many approved blockchain networks enable all users to submit and receive transactions (they are not anonymous or even pseudo-anonymous). In these schemes, parties work together to pursue a shared market procedure with inherent disincentives to commit fraud or even behave as resource-constrained agents (because they can be identified). In unethical conduct, it is merely a judicial process for breach of trust and unethical

business practices. The blockchain system also holds proof of any malicious activity within the systems.

4.8.2 Blockchain-based Consensus Models

A key feature of blockchain technology is identifying the consumer algorithmically and verifying the identity while the consumer publishes the next block. As seen in other distributed systems, this feature and limitation are solved by implementing a consensus model. For permissionless blockchain networks, several publishing nodes are typically competing simultaneously to publish the next block. The competition in publishing the next block arises due to two reasons. One being the computation overhead as the block gets bigger, and the computation becomes complicated. The second reason being the two competing systems mutually distrust each other as they are identified only using their public address. The competition arises from benefit for the self over the other publishing nodes' well-being or even the network itself [94].

With the two reasons above and the two questions, the requirement for a decentralized consensus model needs to be implemented with every blockchain system. Why would a user propagate a block that another user attempts to publish? Also, who solves conflicts when multiple nodes publish approximately the same block? blockchain technologies use consensus models to allow a community of mutually distrustful users to work together within a single network and allow for the reliability of the network health [94].

When a user joins a blockchain network, they accept the system's initial state. Usually, the initial state of the blockchain is called the genesis block. It records the inception of the blockchain in a pre-configured block. Every block that will be published as a reference to a previous block published and validated by the consensus model before the block is publicly available for further transactions. There is no reference to any previous block only for the genesis block, and only one block in the whole of the blockchain operation has such a block. This block is circulated to any node that will join the network. Block parameters such as block size, frequency, and other parameters required for the consensus are recorded in the genesis block. Blockchain networks have a genesis block published.

Consequently, any block published must be added to the blockchain-based on the accepted consensus model chronologically after the genesis block. Regardless of the consensus model, each block must be valid and validated independently by each blockchain user. Leveraging the initial state and the ability to validate each block since the inception of the

network, users can independently decide on the blockchain's current state. Notice that if two legitimate chains were ever introduced to a full node, the default process in most blockchain networks is that the 'longer' chain is regarded as the right one, and the transactions with the longest chain would be adopted. This is because more resources were dedicated to creating the blocks, and chronologically many events have been recorded compared to the other chain. Such scenarios occur when there is a case of split-brain[2] inconsistency in the blockchain consensus model is discussed in detail in their respective algorithms [94]. The following are the summarized properties of a consensus model before a blockchain system is instantiated:

- The initial state of the system is agreed upon (e.g., the genesis block).
- Users agree to the consensus model by which blocks are added to the system.
- Every block is linked to the previous block by including the previous block header's hash digest (except for the first 'genesis' block, which has no previous block and for which the hash of the previous block header is usually set to all zeros).
- Users can verify every block independently.

Practically, in blockchain systems, the software handles everything, and users do not need to be aware of the operational details of the blockchain system. The main benefit of blockchain technology is that there is no need to share the state of the system with a trusted third party — as every user within the system has access to the blockchain network or can check the integrity of the system. All nodes must reach a mutual agreement over time to add a new block to the blockchain. However, some temporary disagreement is possible due to various limitations in operating a computer network, but the consensus models usually allow for such temporary disagreement. The consensus model will work for permissionless blockchain networks even in the presence of potentially malicious users as such users may attempt to disrupt or take over the blockchain. Notice that legal solutions can be used for approved blockchain networks when a user behaves maliciously [94].

Some level of trust between publishing nodes can exist in Permissioned blockchain networks. When this is not the case, the need for a resource-intensive model (computational overhead and operating cost) to decide which member of the network can add the next block to the chain might be necessary. Generally, as the confidence level increases, the need for resources to measure trust generation decreases. For some Permissioned

[2] Split-brain is a computer concept, based on a psychological Split-brain syndrome analogy. This suggests data or availability anomalies arising from the management of two different data sets with similarity in scope, either due to servers in a network configuration or a server-based failure situation that does not interact and synchronize their data. This last case is often widely called a network partition.

blockchain implementations, the consensus view extends beyond ensuring the blocks' validity and authenticity but encompasses the entire system of checks and validations from the transaction proposal to its final block inclusion. Several consensus models and the most common approach to conflict resolution are discussed in the entirety of the following sections. [94]

4.8.2.1 Proof-of-work Consensus Model

A user publishes the next block in the Proof-of-work (PoW) model by becoming the first to solve a computationally intensive puzzle. The solution to this problem is the "proof" that the nodes performed the computation. The puzzle is built so that it is challenging to solve the puzzle, but it is easy to verify whether a solution is correct. Such a computationally intensive model allows the other complete nodes to quickly verify any proposed next blocks and reject any proposed block that did not satisfy the puzzle. [94]

A standard method of creating the puzzles is to allow a block header's hash to be less than a target value. Publishing nodes make several small changes to their block headers (e.g., changing the nonce), seeking to find a hash digest that fits the requirement. The publishing node shall compute the hash for the entire block header for each attempt. Sometimes hazing the block header becomes a computationally intensive operation. Over time the goal value can be adjusted to change the complexity (up or down) to determine how often blocks are released. [94]

For example, Bitcoin, which uses the consensus model Proof-of-work, increases the difficulty of the computation for solving the puzzle every 2016 blocks to influence the block's publishing rate to be about once every ten minutes. The change is made to the puzzle's difficulty level, which increases or decreases the necessary number of leading zeros. Approaches such as this raise the puzzle's complexity by increasing the number of leading zeros. Each answer must be lower than the degree of difficulty – meaning fewer potential solutions exist. It reduces the difficulty level by decreasing the number of leading zeros. This modification is intended to preserve the puzzle's technical complexity and retain the Bitcoin network's central security mechanism. Over time, the computational power available increases, as does the number of publishing nodes, so the puzzle complexity usually increases. [94]

Adjustments to the difficulty goal aim to ensure that no individual can take over the creation of blocks. As a result, the puzzle-solving computations require significant consumption of resources such as energy and computers. There is a move to add publishing nodes to areas where there is a surplus supply of cheap electricity due to the substantial resource consumption of blockchain networks that implement Proof-of-work consensus models. [94]

A significant feature of this model is that the effort put into a puzzle does not influence one's likelihood of solving current or future puzzles because the puzzles are independent. As the computational puzzles are independent, it also means that a user receives a completed and valid block from another user. Incentives are provided for users to discard their current work and start building off the newly acquired block from another user; instead, other publishing nodes will be building off it. [94]

The following is an example of the computational puzzle that is used in the Proof-of-work consensus model, which was explained in [94]. As an example, consider a puzzle where using the SHA-256 algorithm, a computer must find a hash value meeting the following target criteria (known as the difficulty level):

SHA256("blockchain" + Nonce) = Hash Digest starting with "000000"

In this example, the text string "blockchain" is appended with a nonce value, and then the hash digest is calculated. The nonce values used will be numeric values only. This is a relatively easy puzzle to solve, and some sample output follows: SHA256("blockchain0") = 0xbd4824d ... ... ab938 (not solved)

SHA256("blockchain1") = 0xdb0b9c1... ... 3e0a1 (not solved)

SHA256("blockchain10730895") = 0x000000ca1415e0bec... ...9d67587 (solved)

To solve this puzzle, it took 10,730,896 guesses (completed in 54 seconds on relatively old hardware, starting at 0 and testing one value at a time).

In this example, each additional "leading zero" value increases the difficulty. By increasing the target by one additional leading zero ("0000000"), the same hardware took 934,224,175 guesses to solve the puzzle (completed in 1 hour, 18 minutes, 12 seconds):

SHA256("blockchain934224174") = 0x0000000e2ae7... ... db3a81

There is currently no known workaround to this process to find the appropriate nonce value for the target. Therefore, publishing nodes must spend computational effort, time, and resources. Often, the nodes are allowed to publish an attempt to solve this computationally difficult puzzle to assert reward in the blockchain network. The reward is generally in the form of a blockchain network cryptocurrency. The prospect of being rewarded for extending and maintaining the blockchain is referred to as an incentive model or reward system.

Once a publishing node has done the Proof-of-work consensus logic, they send their block to full nodes within the blockchain network with a valid nonce. The recipients' complete nodes check that the new block meets the puzzle criteria, then adds them to their blockchain copy and resend the block to their peer nodes. In this way, the new block can be spread rapidly across the participating nodes in the blockchain network. The nonce verification is simple, as only one hash is generated to test if it solves the puzzle. [94]

Publishing nodes prefer to organize themselves into ”pools” or ”collectives” in a Proof-of-work consensus model. They work together to solve puzzles and share the reward for contributing to the Proof-of-work consensus within a blockchain network. Such a model is made possible in the Proof-of-work consensus model. The computationally intensive puzzles' benefits can be spread between two or more nodes within a group of nodes referred to as pools. Splitting the example program into quarters, each node can take an equal amount of the nonce value range to test as described in [94]:

- Node 1: check nonce 0000000000 to 0536870911
- Node 2: check nonce 0536870912 to 1073741823
- Node 3: check nonce 1073741824 to 1610612735
- Node 4: check nonce 1610612736 to 2147483647

The following result was the first to be found to solve the puzzle: SHA256(”blockchain1700876653”) = 0x00000003ba55d20 ... ... cf16d7f1

This is a completely new nonce, but still, one that solved the puzzle. It took 90,263,918 guesses (completed in 10 minutes, 14 seconds) [94]. Dividing up the work amongst many more machines yields much better results and more consistent rewards in a Proof-of-work model. [94]

Using a computationally difficult puzzle helps combat the ”Sybil Attack”- a computer security attack where an attacker can build several nodes to gain power and control. This form of attack is not only restricted to networks with blockchains. The Proof-of-work consensus model combats this by making the network effect, focusing on the sum of computational power combined with a lottery system. The nodes with the most potent hardware increase the chance but do not guarantee it. Finally, the Proof-of-work consensus model combats network identities, which generally cost less to create. [94]

4.8.2.2 Proof-of-stake Consensus Model

The Proof-of-stake (PoS) model is based on the idea that the more stake a user has invested in the system, the more likely they will want the system to succeed, and the less likely they will want to overturn it. A stake is also a sum of cryptocurrency that the blockchain network user has invested in the system. The means of investment can consist of different methods, such as locking it into a specific form of transaction, sending it to a particular address, or keeping it in a specific wallet that can be publicly validated by any other participating system. Once staked, it is generally impossible to spend the cryptocurrency any more. Proof-of-stake blockchain networks consider the amount of stake a user has committed as a critical parameter for publishing new blocks. Thus, a blockchain network user's probability of

publishing a new block depends on the ratio of their committed stake into the system versus the amount of the total staked cryptocurrency blockchain network. [94]

As found in the Proof-of-work detailed in the earlier section, there is no need to perform resource-intensive computations in the Proof-of-stake consensus model. Intensive computations are computations that involve time, energy, and processing power. As this consensus model requires fewer resources than the Proof-of-work, some blockchain networks have opted to forego a block development reward. Blockchain systems with such a consensus model are designed to allocate all the cryptocurrency to users, rather than creating new cryptocurrency at a constant rate, thereby devaluing the cryptocurrency itself. The incentive for block publication in these systems is usually the earning of the user's transaction fees paid by the user. [94]

Methods of how the stake uses the blockchain network may vary. We address four approaches: random stakeholder collection, multi-round elections, coin aging systems, and delegate systems. New blocks are more likely to be published by users with more stake whatever the exact approach is. If selecting a block publisher is a random choice (sometimes referred to as chain-based stake proof), the blockchain network can look at all stakeholder users and select between them based on their stake ratio to the total amount of stakeholders cryptocurrency. So if a person had 42 percent of the overall stake in the blockchain network, they would be selected 42 percent of the time; those with 1 percent would be selected 1 percent of the time. [94]

Complexity is added when choosing block publishers is a multi-round voting system. This form of consensus model is called Byzantine fault tolerance, Proof-of-stake [94, 95]. The blockchain network selects multiple users staked to build proposed blocks. All the users staked will then cast a vote for a proposed block. There could be several rounds of voting before a new block is determined. This method helps all the staked users vote for every new block in the block selection process. [94]

When the block publisher's option for publishing into the blockchain exists through the age of a coin system, then the consensus model is referred to as a coin age Proof-of-stake. In this case, staked cryptocurrency has an age property. After a certain amount of time (such as 30 days), the staked cryptocurrency can count towards the owning user being selected to publish the next block. Instead, the staked cryptocurrency has its age reset and cannot be used again until the appropriate time has passed. This method helps users with more stake to publish more blocks but not control the system – because they have a cool-down timer attached to each cryptocurrency coin counted against block formation. This cool-down timer helps in allowing other participating systems a chance to publish blocks. Older coins and larger

groupings of coins will increase the likelihood that the next block will be issued. Traders can hoard aged cryptocurrencies to benefit from the systems consensus model. To avoid such behaviour from the participating systems, the probability of winning is usually set to a predefined limit. [94]

Once the selection of block publishers is performed through a delegate method, users vote for nodes to become publishing nodes – thereby generating blocks. The voting power of the nodes in the blockchain network users depends on their committed stake. The stake that a user commits is that the greater the stake, the higher the weight of the vote. Nodes that get the most votes become publishing nodes and can validate blocks and publish them. Blockchain network users may also vote against a publishing node formed to exclude them from the publishing node group. Voting to publish nodes is ongoing, and the remainder of a publishing node can be quite competitive. The threat of losing the status of publishing nodes for acting maliciously and the rewards and constant reputation encourages publishing nodes not to act maliciously. Furthermore, members of the blockchain network vote for delegates interested in blockchain governance. Delegates will recommend updates and enhancements to be voted on by members of the blockchain network. [94]

It is worth noting that some evidence of stake algorithms can give rise to a problem known as ”nothing on stake.” When a temporary ledger dispute arises, i.e., if several competing blockchains were to occur at any stage, a staked consumer might operate on any such competing chain. As mentioned earlier, the action does not have consequences because they are considered different blockchain systems where the users participate in either one of the split chains. The consumer staked can do this as a way to improve his chances of receiving a reward. It may allow multiple blockchain branches to expand for extended periods without reconciling into a single branch. [94]

Under Proof-of-stake systems, the ”rich” can easily stake more of the digital assets, earning themselves more digital assets; however, to obtain the majority of digital assets within a system to ”control”, it is generally cost prohibitive. [94]

4.8.2.3 Round Robin Consensus Model

Round Robin is a model of consensus that some approved blockchain networks use. In this consensus model, nodes take turns in block formation. The Round-robin model of consensus is well proven and widely used for scheduling in a distributed systems architecture. One such example of a distributed systems architecture is in load balancing application servers to improve the distributed servers' reliability. In scenarios where a publishing node cannot publish its block during its turn, systems can have a time limit

to allow accessible nodes to publish blocks so that inaccessible nodes do not bring block publishing to a halt in the blockchain system. This model ensures that one node does not create the majority of blocks. It benefits from a straight-forward approach, lacks cryptographic puzzles, and needs low computation and energy overhead. [94]

As trust among nodes is required and enforced using contractual means by the consortium participants, round-robin does not work well in permissionless blockchain networks. Most of the cryptocurrencies use permissionless blockchain. Therefore, using a round-robin consensus model in the cryptocurrency domain may not suit all its requirements. The limitation in permissionless blockchain networks is that malicious nodes could add additional nodes to increase their odds of publishing new blocks. They may use this in the worst case to subvert the blockchain network's proper functioning. [94]

4.8.2.4 Proof of Authority/Proof of Identity Consensus Model

The consensus model called proof of authority (also referred to as proof of identity) relies on the publishing nodes' partial confidence through their known link to real-world identities. Publishing nodes must have confirmed and verifiable identities within the blockchain network (e.g., identification of documents checked and notarized and placed on the blockchain). The theory is to publish new pieces. The publishing node stakes its identity/credibility. Blockchain network users directly influence the credibility of a publishing node based on the actions of the publishing node. Publishing nodes can lose credibility by behaving in a way that the blockchain network users disagree with, just as they can gain reputation by acting in a way that the users of the blockchain network embrace. The lower the credibility, the less likely a block will be written. Hence sustaining a good reputation is in the interest of a publishing node. This algorithm only applies to approved high confidence blockchain networks. [94]

4.8.2.5 Proof of Elapsed Time Consensus Model

In the proof of the elapsed time (PoET) consensus model, each publishing node demands a waiting time inside their secure hardware time source until it is determined to be ready for block publication. The stable time source for the hardware will produce a random wait time and return it to the publishing node's program. Publishing nodes take their assigned random time and remain idle for that random period. When a publishing node wakes up from the idle mode, it generates and publishes a block to the blockchain network,

alerting the new block's information to the other nodes. Any publishing node that does not have a block to publish is idle and can stop waiting to follow the process with the new cycle start. [94]

This model involves ensuring that the participating nodes use a random period. The random period is necessary because any malicious actor would not predict the wait times of other systems. When a malicious agent in the network can predict other nodes' waiting times, it can dominate the system by publishing blocks without allowing different systems to participate. The PoET consensus model also needs to ensure that the publishing node waited for the actual time and did not start early. Executing software in a trusted execution environment found on some computer processors (such as Intel's Software Guard Extensions5, or AMD's Platform Security Processor6, or ARM's TrustZone7) helps the software to adhere to these strict requirements for the consensus models to perform. [94]

Execution within these stable environments where verified and trustworthy software can run and can not be altered by external programs even during runtime is critical for the PoET consensus model to function. Any publishing node in the blockchain network would query the service in a secure hardware environment for a random time. Followed by the query, the software will wait for this time to pass. While the CPU is waiting for the timeout to publish the next block, it can dedicate the resources to other computational tasks. After waiting for the random time allotted, the publishing node requests a signed certificate that the publishing node was waiting for the randomly allocated time. The certificate is then released along with the block by the publishing node. [94]

4.9 Decentralized Consensus Protocols

Blockchains are tamper-evident and tamper-resistant digital ledgers implemented in a distributed manner (i.e., without a central instance) and usually without a central authority (i.e., a bank, corporation, or government). At their basic level, they require a group of users to record transactions within that group in a shared ledger [94]. That way, no transaction can be modified once published under the regular operation of the blockchain network. In 2008, to build modern cryptocurrencies, the blockchain idea was combined with many other innovations and computing concepts: decentralized cash secured by cryptographic processes rather than a central repository or authority. The first such cryptocurrency based blockchain was *Bitcoin*. [94]

Blockchain technology is the foundation of modern cryptocurrencies, so named for the heavy use of cryptographic functions. Users use both public

and private keys to sign and transact safely inside the system digitally. For cryptocurrency based blockchain networks using mining (see Sec. 4.8.2.1), users can solve puzzles using cryptographic hash functions in hopes of being rewarded with a set cryptocurrency number [94]. Blockchain technology can, however, be more commonly applicable than cryptocurrencies. In this work, we focus on the use case of cryptocurrency, which is the primary use of technology today; however, there is increasing interest in other sectors. [94]

Blockchain architecture is also designed for a particular purpose or function. Example functions include cryptocurrencies, smart contracts (software implemented on the blockchain and executed by computers running the blockchain), and distributed ledger systems among companies. There has been a constant flux of blockchain technology developments, always announcing new platforms – the landscape is continually changing [94]. In this work, we explore using blockchain at its bare minimum to develop a byzantine fault-tolerant consensus for decentralized systems. The consensus is used as part of a software stack developed as a context broker for heterogeneous high-power, high data-rate systems. [94]

The use of blockchain technology is not a magic bullet. Some problems need to be addressed, such as managing malicious users, how restrictions are implemented, and the implementation limitations. Among the technical problems that need to be tackled, organizational and governance concerns impact the network's behavior [94]. In this work, we meticulously explore options for developing a decentralized consensus scheme for a context broker. Since all of the context broker requirements are met, we choose to develop a blockchain network that is suitable for this application. Furthermore, blockchain is used only as a decentralized consensus system developed and demonstrated in Sec. 4.11.2 and 4.11.3.

Industry 4.0 and the concept of socially networked industry define that the entities in an industry are networked and can communicate with each other to collaborate for performing tasks autonomously [75]. dezCom is an architecture for **de**centrali**z**ed **com**munication and messaging. For a truly decentralized industrial process, the communication between the entities that take part in the process should also communicate in a decentralized manner. Even though the execution algorithm is decentralized in nature, it is not truly decentralized when running on a centralized communication architecture, such as MQTT or Orion context broker. Therefore, the effort to abstract the communication layer with a purely decentralized communication framework is implemented and presented in the DezCom communication stack. Decentralization using blockchain increases the broker's availability and increases the security of data in a trust-less network. There are many implementations of decentralized robots based on blockchain [76, 77]. DezCom contributes to developing a first-ever decentralized industrial supply chain messaging system based on

mining-less blockchain, emphasizing messaging using assets in a socially networked industry. Such a stack also provides mitigation against a central point of failure and emphasizes on edge computing. The following three sections 4.9.1, 4.9.3, 4.9.2 study the three candidates for decentralized consensus protocols developed as open-source software in various blockchain networks. After evaluating the application problem fit, one of the candidates is selected, and the DezCom system is designed with a decentralized byzantine fault-tolerant consensus protocol.

4.9.1 Merkle tree

Merkle Trees are one of the reasons blockchain technology has managed to be so successful today. It is the underlying concept for every blockchain technology where the whole blockchain is not necessary to validate new blocks. It helps form a hash tree that can be leveraged to easily associate newly joining nodes with an extended network overhead to download the whole blockchain [96]. Since every other block references the preceding block, the blockchain becomes tamper-evident and resistant. In the rest of this section, we will understand the functionality of the Merkle tree. [94]

A Merkle tree is also referred to as a hash tree. It is a tree of hashes[3] in which the leaves are hashes of data blocks in a file or set of files. Nodes further up in the tree are the hashes of their respective children. For example, in the picture hash, 0 is the result of hashing the concatenation of hash 0-0 and hash 0-1. That is, hash 0 = hash(hash 0-0 || hash 0-1) where || denotes concatenation [96, 97].

Most hash tree implementations are binary (two child nodes under each node), but under each node, they can just as quickly have several more child nodes. A cryptographic hash function like SHA-2 is typically used for the hashing. If the hash tree needs to only protect against unintended damage, unsecured checksums like CRCs can be used [97].

There is a top hash (or root hash or master hash) at the top of a hash tree. In most cases, before downloading a file to a p2p network, the top hash is acquired from a trusted source, such as a friend or a website known to have good file recommendations to download. The hash tree can be received from an untrusted source, just like any peer in the p2p network, when the top hash is available. Then, the hash tree obtained is verified against the trusted top hash, and if the hash tree is corrupted or false, another hash tree will be tried from another source until the program finds one that matches the top hash [96, 97].

[3] an unidirectional function that transforms an arbitrarily sized data input into a unique, fixed-size representation

4.9.2 Tangle

In this section, we will discuss the innovative blockchain technology developed with the IoT industry in focus. IOTA is a distributed ledger technology that uses a different form of a graph compared to other blockchain technologies. The most commonly used graph data structure is the Merkle tree or hash tree. Here, IOTA uses a different kind of tree called the Directed Acyclic Graph, as illustrated in Fig. 4.12. In search of mining less or an effortless mining based distributed ledger system, we study the benefits, functions, and features of IOTA. Blockchain network fees are not easy to get rid of because they act as an incentive for block developers [98]. This leads to yet another problem with current cryptocurrency technology, namely the system's heterogeneous design [98]. The network consists of two distinct classes of stakeholders, those who request transactions and those who authorize transactions [98]. This system's design forces inevitable discrimination for some stakeholders, which creates conflicts that cause all elements to spend resources on conflict resolution. [98]

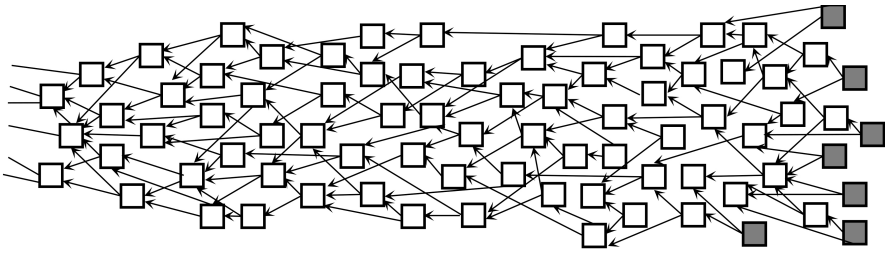

Fig. 4.12 Visual representation of DAG-Based Tangle ledger [98]

IOTA is a cryptocurrency that implements the tangle based distributed ledger algorithm. The network is composed of nodes; that is, nodes are entities that issue transactions and verify them. The tangle's core premise is the following: Users will work to accept other transactions to be granted a transaction in the distributed ledger. This concept of approving previously available transactions to create new transactions is illustrated in Fig. 4.12. Thus, users who issue a transaction contribute to the security of the network. The nodes are presumed to test if the authorized transactions do not conflict. If a node considers that a transaction conflicts with the tangle history, either directly or indirectly, the node does not approve the transaction in dispute [99]. Even though tangle cannot be used as a consensus platform, to a certain degree, the nodes agree on the transactions which provide consensus. One of the drawbacks is that the blockchain database is not global. Instead, there is a Directed Acyclic Graph (DAG) called the tangle,

where the tips of the transactions are growing in all directions. The growth of the DAG tree, i.e., the tangle blockchain, depends on the nodes proposing the transactions. [98]

The transactions are stored in the site[4] set of the tangle ledger. The tangle's edge set is obtained in the following way: when a new transaction arrives, it must approve two previous transactions. Approving two previous transactions is the simplest approach. However, there are also similar systems where transactions must approve *k* other transactions for a general k≥2, or have an entirely different set of rules. Directed edges represent these approvals. If there is no directed edge between transaction A and transaction B, but there is a directed path of length of at least two paths from A to B, we say that A indirectly approves B [98, 100]. There is also the "genesis" transaction, which is accepted by all other transactions either directly or indirectly. The genesis here is defined as follows. There was an address at the beginning of the tangle, with a balance that included all the tokens. Such tokens were sent to many other "founders" addresses by the genesis transaction. Let us emphasize that the genesis transaction produced all of the tokens. No tokens will be produced in the future, and no mining will occur in the sense that miners earn monetary rewards "out of thin air." [98, 100]

When a transaction earns more approvals, it is approved with a higher degree of trust by the network. In other words, a double-spending transaction would be difficult to get the system to accept. It is important to remember that there are no enforced rules on selecting which transactions a node approves. Instead of that, if many nodes follow some "reference" rule, then it is better to stick to a rule of the same kind for any fixed node. This seems to be a rational assumption, especially in the sense of IoT, where nodes are specialized chips with firmware pre-installed [98]. In order to issue a transaction, a node does the following:

- The node chooses two other transactions to approve according to an algorithm [98]. In general, these two transactions may coincide.
- The node checks if the two transactions are not conflicting, and does not approve conflicting transactions [98].
- For a node to issue a valid transaction, the node must solve a cryptographic puzzle similar to those in the Bitcoin blockchain. This is achieved by finding a nonce such that the hash of that nonce concatenated with some data from the approved transaction has a particular form [98]. In the Bitcoin protocol, the hash must have at least a predefined number of leading zeros [98].

With this understanding of creating transactions and approving other transactions, we can clearly understand that the blockchain is asynchronous. This

[4] sites are transactions represented on the tangle graph

also means that valid transactions created and validated by part of the system are not known to the rest of the system at any given point in time. Even though it saves memory and propagates the network to generate more transactions with less overhead, at any point in time, all nodes in the network do not agree to all transactions in the system. [98]

The IOTA cryptocurrency and tangle are an excellent solution for applying blockchain applications on low-power, low-data rate systems. There are certain limitations on the types of systems that are capable of creating signed transactions. Certain micro-computers are not capable of approving transactions due to constraints on their computing resources or energy. [98]

4.9.3 Tendermint

Tendermint uses hash-linked batches of transactions to provide Byzantine fault-tolerant state machine replication. These batches of transactions are called blocks. Therefore, Tendermint can be described as a blockchain application with a decentralized byzantine fault-tolerant consensus. [101]

Every Tendermint block has an unique index. The index is referred to as Height. In the blockchain, Height is monotonic. A known set of weighted validators commits each block. In this validator collection, membership and weighting can change over time. Tendermint guarantees the blockchain's protection and liveliness as long as less than 1/3 of the Validator set's total weight is malicious or faulty. [101]

A Tendermint commit is a compilation of signed messages of more than 2/3 of the current Validator set's total weight. Validators each take a turn to propose and vote on those proposed blocks. The block is deemed committed to the blockchain database until appropriate votes are issued. These votes are used as proof that the previous block was committed. In the next block, transactions can not be included in the current block because that block has already been formed. [101]

Once a block has been committed, it can run against an application. For every transaction in the block, the application returns results. The application will also be able to return changes to the validator set and a cryptographic digest of its current state. [101, 102]

Tendermint is designed to verify and authenticate the current state of the blockchain efficiently. To this end, it integrates cryptographic commitments in the block "header" as metadata. Such information contains the block headers (e.g., transactions), the block committing validator set, as well as the different results the client produces. Nevertheless, note that the block

execution happens only after a block has been committed. Therefore, the results of the application can be included only in the next round. [101]

Also, note that the transaction results and the validator set are never included explicitly in the block. Only its cryptographic digests (Merkle roots) are included in the block. Hence, verification of a block requires a separate data structure to store this information. This data structure is called the State in Tendermint. Block verification also requires access to the previous block. [101, 102]

Tendermint is broadly akin to two software categories [101]. The first category consists of distributed key-value stores, such as Zookeeper and consul store, which use consensus rather than BFT [101]. The second category is known as ”blockchain technology”, which comprises both cryptocurrencies such as Bitcoin, Ethereum and alternative distributed ledger implementations such as Burrow from Hyperledger [101].

The Tendermint Core (the ”consensus engine”) connects with the application through a socket protocol that meets ABCI requirements. Tendermint can decompose the blockchain architecture by providing a straightforward API (i.e., the ABCI) between the implementation and the process of consensus [102]. The ABCI consists of three primary message types, which are delivered to the application from the core. The program responds with messages with corresponding responses [102]. One program can have multiple ABCI socket connections. Tendermint Core provides three ABCI links to the application; one to validate transactions while broadcasting in the mempool[5], one to block proposals on the consensus engine, and one more to query the application state. It is apparent that application designers need to design their message handlers very carefully to create a blockchain that does something useful, but this framework provides a starting point [101, 102].

4.10 DezCom: A Decentralized Context Broker

Due to the nature of the consensus protocol and the analogous features widely used in distributed systems, Tendermint is selected as the prospective candidate for developing the DezCom context broker. Tendermint also provides an API that communicates using a web socket. Such an interface to the ABCI allows for any user program written in any language to openly integrate into the blockchain without considering the consensus and blockchain [102]. The context broker does not necessarily

[5] A mempool (a contraction of memory and pool) is a cryptocurrency node’s mechanism for storing unconfirmed transactions. It acts as a sort of waiting room for transactions that have not yet been included in a block.

require a blockchain but needs to be decentralized with certain security features to safely run industrial applications. In Sec. 4.7, we systematically explore features of blockchain provided by Yaga et al. [94] as a guide to evaluating whether blockchain is the right software component for the application that is being developed.

In the following Sec. 4.10.1, a system architecture is proposed, including the decentralized consensus provided by Tendermint. Followed by the system architecture, the proposed architecture is implemented in Sec. 4.11, where a use case is introduced that is suitable for exploring the features of such a decentralized context broker. The implementation and the results follow the use case discussed in this section.

4.10.1 Systems architecture

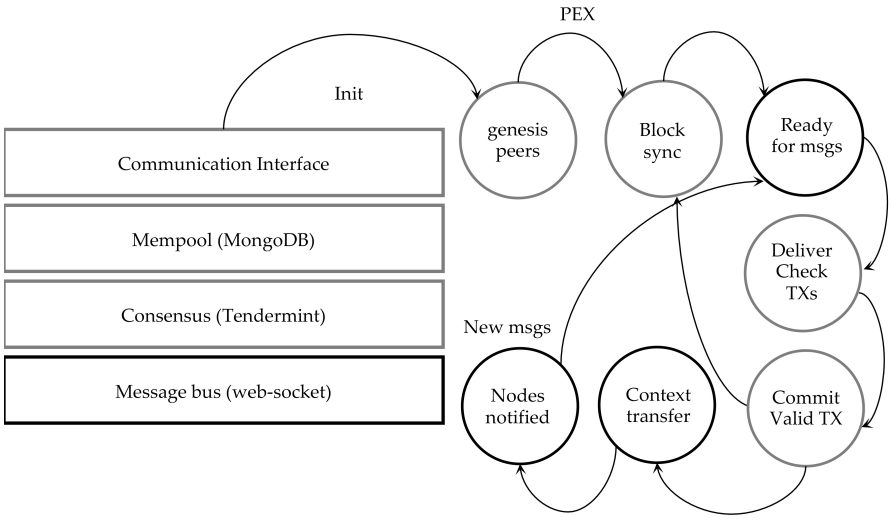

Fig. 4.13 State machine of the DezCom stack with messaging steps [56]

The implemented decentralized context broker which is comprised of three main layers is shown in Fig. 4.14 (left) with a ***decentralized consensus protocol*** Tendermint. These layers are a modular black-box style implementation to keep the flexibility on the consensus protocol and it seamlessly weaves into the networking stack of the deployed system. The stack has a *communication interface*, which is initially implemented with the well-known TCP/IP stack with IP networking. This communication

interface can be changed, but the basic services such as reachable, addressable nodes with functions for discovery need to be provided [56]. Any low-power wireless sensor network with a 6LoWPAN networking stack can also be used to run this context broker if required computations resources are available. Even though 6LoWPAN based low bandwidth networking can be used as a communication interface, there are specific minimum requirements in terms of the computational resources. The few computational resources that do not comply with implementing this system for signing the transactions might not be suitable for deploying the context broker in low-power, low data-rate systems. A memory pool or *mempool* is used to store all the incoming transactions (UTXO) through the communication interface. The mempool is also where the data is stored after consensus. It can be compared to MongoDB in FIWARE; the mempool is also implemented using MongoDB in the networking stack. This is where the unresolved data is stored before being validated by the consensus protocol. The consensus protocol is also replaceable, where any decentralized consensus algorithm can be used. In this reference implementation, Tendermint is used for the byzantine fault-tolerant nature of consensus and the mining-less blockchain protocol [101]. Another reason to use this blockchain-based consensus is that there is no mining required in the Tendermint protocol [103] [56].

There is a socket interface where the events of the consensus protocol are triggered by the *message bus*, which interfaces with the application to perform *CREATE*, *UPDATE* transactions as wells as to query time series data. Since the consensus is ledger-based, there is no *DELETE* operation. Instead, an update operation should be made on the data to deem it invalid, or in a context broker case, the process state needs to be changed to completed to mark the end of a process [56]. In Fig. 4.14, the consensus protocol of Tendermint is shown in simplified steps as used in the DezCom framework. Nodes connected to a particular node are called *peer nodes* in the P2P network where P2P network discovery happens using the Peer Exchange protocol (PEX) as used in the bit-torrent networks. In industrial mobile robots, all the field robots are connected to the FMS as peers. At the same time, the WMS connects to low-level hardware such as the PhyNodes[10] using an edge router as a client, which is Raspberry-PI based. Any system with necessary hardware requirements can host a node for consensus or connect to a node using the API interface.

The decentralized consensus happens in multiple rounds, where the process in deciding on the next block (at some height H) is composed of one or many rounds[101]. NewHeight, Propose, Prevote, Precommit, and Commit are the states/steps (S) in a state machine of every round (R) [101]. At each height of the blockchain a round-based protocol is run to determine the next block which is depicted in Fig. 4.14. Each round is composed of three steps,

as shown in Fig. 4.14, Propose, Prevote, and Precommit, along with two special steps, Commit, and NewHeight. Valid transactions become available on the message bus after the Commit step is completed. In the optimal scenario, the order of steps is NewHeight to (Propose —>Prevote —>Precommit) to Commit to a new height, and so it goes on [101] [56].

Logistics applications are interfaced using the message bus as an ***application interface***, where the ABCI of Tendermint is implemented. Events and REST API are the two ways to communicate with the Tendermint protocol. There are 15 events available from the web socket, of which the valid transactions are most important for the context broker. The

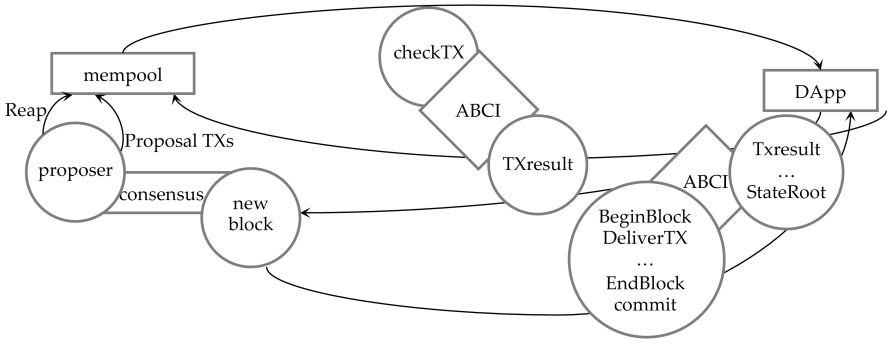

Fig. 4.14 Tendermint [101] consensus protocol as used in DezCom [56]

DeliverTX message is where each transaction in the blockchain is delivered with the message to be verified to all the connected peers as validators [101]. A validated transaction needs to update the application state — by binding a value into a key-value store or updating the UTXO database. The CheckTx message is similar to DeliverTx, but it is only for validating transactions [101]. The Commit message is used to compute a cryptographic commitment to the current application state to be placed into the next block header. This simplifies the development of secure, lightweight applications, as Merkle-hash proofs can be verified by checking against the block hash, and that a quorum [101] signs the block hash. From a context broker perspective, assets and the payload the assets carry are more critical, and integrity checks can be made by checking against the block hash. In this context broker, an assets-based communication model is preferred over a topic-based communication. The reason for such a communication model is detailed in Sec. 4.11.1 using an example from the automotive manufacturing industry [56].

4.11 Implementation of DezCom

In this section, a use case is defined where heterogeneous systems are connected to DezCom. A 6 node testnet DezCom stack is deployed, and the results are analyzed for the framework, which is presented in the following sections.

4.11.1 Use case

The life cycle of customized product manufacturing tracked through the whole supply chain with support for various business services, the goal of a socially networked industry [75], is the use case. The use case takes the challenge of connecting flexible production stations in a decentralized approach, as defined in SMARTFACE [104]. The messaging approach is addressed using assets, i.e., the products in the production instead of production stations directly or topics as in a publish-subscribe model. For example, consider a customized pre-order system of electric vehicles using a web-interface. When an order is placed, an asset is created in the DezCom network. This created asset is then transferred between workstations with respective messages that every production station can carry out their physical transformation to the manufactured product. By the end of the production, every message pertaining to the asset's production is recorded in the blockchain [56].

4.11.2 Stack implementation

The DezCom testnet network is deployed with cloud servers in 4 major cities and two local instances in a logistics scenario. The cloud servers acted as validators in the experiment implementation infrastructure. The local nodes ran an instance of Tendermint and connected to the cloud servers using the PEX protocol. Nodes that need to communicate using DezCom hosted a Tendermint instance and communicated to the local instance. Nodes that did not meet the minimum requirements for running a Tendermint instance (e.g. RPi) were using the REST API interface to connect to local instances. The messaging is relayed to other servers using Tendermint after the transaction is committed to the blockchain. An asset is used as a messaging topic in DezCom where an asset is initialized using a *CREATE* transaction in the blockchain. The created asset is issued with a unique hash ID, and it is recorded in the blockchain as a transaction. The

assets can be queried using this hash ID from the blockchain. The data model for communication is not in this experiment; here, we try to evaluate DezCom as a reliable context broker where messages with a constant payload size are sent around for testing purposes. Topics play an essential role in providing context to the messages. In contrast to topics for context messaging, when using assets-based context management, the asset can be reassigned to a different process step due to the possibility of collaborative machines. When such reassignment is performed, the asset with the information available dictates the decision of movement within the handling facility rather than a process manager service that controls the actions of the mobile robots. Therefore, if any node in the network was to be lost, the consensus did not grind to a halt. Lost nodes were not able to perform the necessary transactions to the asset, which triggers another station. The measurements were made on this deployed testnet, and their results are discussed in the following Sec. 4.11.3.

4.11.3 Results

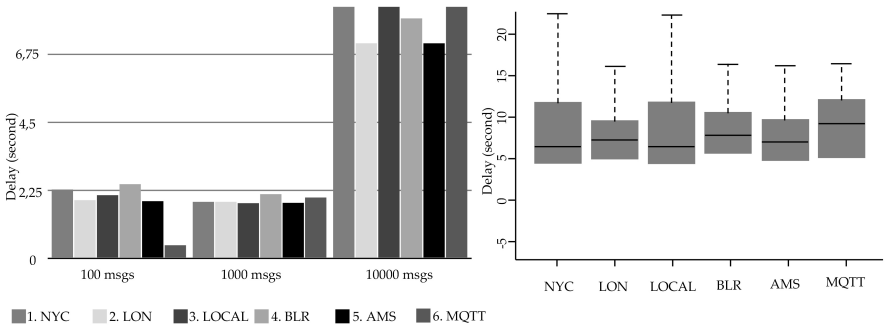

Fig. 4.15 Comparing message reception at various locations with MQTT, box plot with maximum delay over all trails [56]

Performance measurement was creaated between MQTT and DezCom where 100, 1000, and 10,000 messages were sent with 4 cloud servers acting as validators for DezCom. Each testnet server has a 2x vCPU and 2GB of RAM hosted in 4 data centers in New York City (NYC), London (LON), Amsterdam (AMS), and Bangalore (BLR) for location transparency. The local instances were run on an actual robot that otherwise ran the MQTT client during operational tests for industrial scenario validations. The results, as plotted in Fig. 4.15 show that the context broker has constant

delay comparing the runs between 100 and 1000 messages. This delay was because of the decentralized consensus process before the messages were delivered. MQTT outperforms at a run where 100 messages were sent.

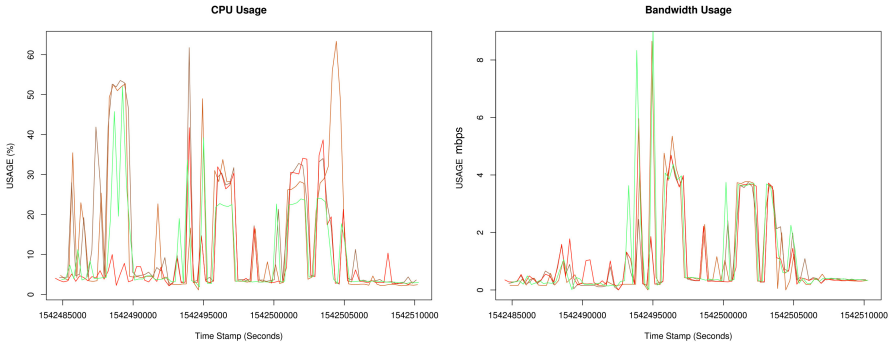

Fig. 4.16 Resource usage for cloud servers with Tendermint consensus [56]

During the trial run with 1000 messages, where MQTT performance decreases, DezCom has the same performance as 100, which was due to the availability of messages committed into a single block. Another trial was run with 10,000 messages where DezCom performed as good as MQTT. MQTT stalled when a run with 100,000 messages was executed. Whereas the DezCom local instance crashed due to inconsistencies in the socket interface driver of ABCI, the validators were not overloaded during the 100,000 request run, which can be inferred from the cloud service providers' data 4.16. The average time for preparing and sending the message is 18 ms. This was measured at every test and averaged throughout the tests. This number depends on the hardware where the messages were signed with the cryptographic keys. In this case, robots running ROS on a 4 core and 8 GB RAM processors were used.

4.12 Conclusion

Decentral systems are fundamental building blocks for self-organizing, collaborating machines. In the case of a socially networked industry, where the field systems and the business logic have to organize themselves in a network and function collaboratively, truly decentral systems are required. To build truly decentralized industrial applications, **DezCom**, a decentralized context broker was conceptualized as a communication framework [56]. The implemented communication stack was deployed in a robotics application for goods-to-person picking scenarios where the FMS,

WMS, and the robots communicate using DezCom by transactions on the blockchain [56]. These transactions are triggered as events, and every task is completed in stages by acknowledging that every stage is made as an update transaction to the data. This proof-of-concept implementation is for partners to organize in a network and communicate in a socially networked industry. The implementation has no single point of failure and almost no bottlenecks in scaling [56]. These features were demonstrated with requirements such as location transparency, N-way scaling, and fault tolerance, which were not existing in centralized brokers. A feature to prioritize the set of validators for specific assets and transactions improves inherent latency due to the geographically collocated nodes with varying round trip times. The validator preference set as a field on the transaction asset data structure is suggested for future releases. The validator preference setting will create and validate the transactions that are also physically responsible stations in the production/warehousing facility. Additionally, a consortium based server that can provide secondary support on validating the transactions is suggested for production deployment. Moreover, the ABCI currently used for the proof-of-concept was a Tendermint testnet in the future DezCom release, where transactions should be possible between two or more Tendermint blockchains. Interoperation of Tendermint chains will provide a proper trust-less integration between networks, i.e., industries in the supply chain. Finally, to use this decentralized messaging system, semantics for the transaction must be developed to generalize the process stages between participating entities using process interpretation techniques. The model is validated before propagating transactions into the chain.

Notes on production deployment: A load-balancer can proxy the connections between the cloud servers, which reduces any direct load to a specific server. The cloud servers for tests were deployed with an SSL terminated Nginx web-server running on an Ubuntu server. PEX protocol will gossip about the other server locations within Tendermint [56]. Therefore, one common address with the genesis file should be copied to all the node instances and allowed enough time to synchronize with the blockchain. In the case of an industrial implementation throughout the supply chain, every location should host a validator node to improve the stability in messaging and reduce the context broker's geographic dependency.

Chapter 5
Designing Decentralized Systems

Decentralization is the process of dispersing power away from a central authority.

In this chapter, three use cases are evaluated, for which the proposed solutions are suitable. A part of the proposed solution was deployed in a similar environment to evaluate the use cases and the requirements of implementing such a system. The three use cases are detailed in the following two sections 5.1, 5.2 and 5.3. Sec. 5.1 explores the use case of space applications with the help of TESSERAE. Sec. 5.2 explores the use case of logistics and distributed robotics where the benefits of decentralization are also described. In Sec. 5.3, the use case of deploying both the decentralized communication systems developed as a single system is discussed. This use case is where specific applications do not require decentralization at the lowest levels, but the edge nodes need to be resilient and always available. Beyond being always available and ready for communication, it is also necessary for such systems to agree with the system's global state.

The last Sec. 5.4 in this chapter details the best practices and the design requirements that will fit a system design to evolve from a centralized or a distributed network into decentralized networks.

5.1 Space Applications

The TESSERAE self-assembling space structures project [105, 106] is one of the examples of an in-orbit, collective maneuver space system necessitating distributed shared-state awareness among self-assembling nodes (Fig. 5.1). Each tile that comprises the geodesic dome shell structure is augmented with a sensor node that participates as part of a decentralized sensor network. While initial work focused on the use of the BLE standard for star-topology or mesh-based communication between tiles, we are now exploring the use of the communication architecture described below to share IMU data, physical and system state information between tiles. This

will be used to actuate many electromagnetic interactions in parallel as LIDAR-based sensing indicates that tiles are approaching one another during the structures' distributed assembly process [105, 106]. Although most close-range distributed spacecraft protocols may assume that all others can hear each radio, there are scenarios where the large tiles may shield reception from others, necessitating networked operations. Here the communication system developed reduces the probability of shielding as the synchronous broadcast forms a fully connected, routing-less multi-hop mesh network.

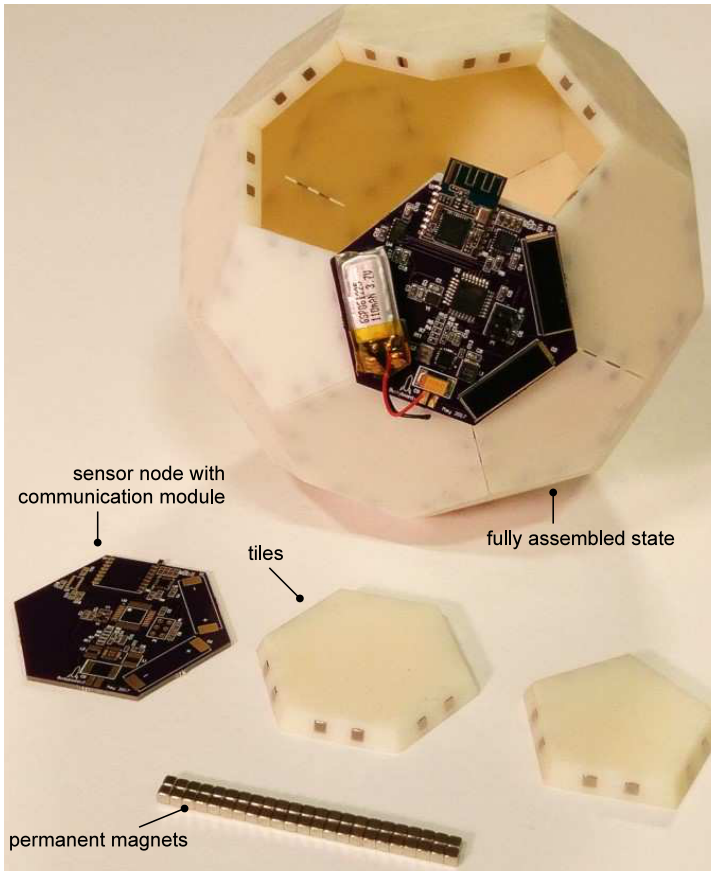

Fig. 5.1 TESSERAE prototype with the sensor node [105, 106]

Data consensus must be achieved to ensure an optimized use of the electromagnets and keep the power budget to a minimum while tiles are assem-

bling into the final structural shell. In this context, we are exploring five parameters described in more detail below:

Aggregate functions for duplication-sensitive holistic data representation across nodes.

Network-wide Agreement for state-awareness (notably proximity) among neighbor nodes.

Atomic broadcast for iterative, two-way propagation of state between a leader node and client nodes.

Reliable Data Dissemination for detecting packet loss and addressing loss recovery.

Modular Communication Patterns allowing for flexibility in the communication topology and protocol being used (star, mesh, one-to-many, one-to-all, all-to-all etc.) along with 6LoWPAN or BLE.

For self-assembly in space, the networking application should provide ad-hoc capabilities, where a new network is created and allows all other nodes in the vicinity to join the network. When a node is faulty or leaves the network due to an event that was not planned, these events must be mitigated by self-healing capabilities. In the case of TESSERAE, swarm behavior is transient during assembly. Therefore, nodes require low latency data dissemination as well as adaptive replication strategies during interaction to enable precise control of interacting elements or guarantees in information propagation [55].

TESSERAE as a use case for the concept of *Decentralized Brains* presented here, the functional stages of assembly as illustrated in Fig. 5.2 can be seen from the networking perspective [55]. The first tile that is deployed or turned on will become the inherent leader starting broadcasts and waiting for nodes to discover [55]. At this stage of assembly, the broadcast round consists of the state from the leader node [55]. As the tiles are deployed and reach the free floating state, they discover the network and start receiving the shared states from the leader [55]. As per the requirements of TESSERAE [106], the required broadcast is 22 bytes sensor data transfer; here a reliable data-rate of 124 bytes per replication round, excluding the physical layer header and the replication header, is available [55]. The data-rate is increased by decoupling the meta-data and serialization information for data replication using the message type bit in the CMD field; this maximizes the data replication since the same data structure persists with consistency more frequently than the changes to the information itself in collaborative control maneuvers [55].

In addition to the TESSERAE self-assembly swarm, we note an applicability for this work to other space applications, such as a space-fed phased array antenna swarm. Here, each element of the array is free to roam within a certain area, while compensating for position error via selectively delaying and amplifying signals to generate a coherent radio signal [107]. The communication architecture should facilitate relative position tracking

between swarm elements and the necessary variable-delay updating and re-synchronization of timing between emitters to achieve coherently cooperating RF elements [55]. This is in contrast to a fixed-array antenna, where elements are held rigidly along a plane and can use a standard delay based on fixed positions along the antenna element line [55].

In large-scale, low-power networks, the *Decentralized Brains* concept is achieved using data replication with variably participating nodes, controlled by the *cluster-differentiate setting* in the CMD field of the payload. If the nodes are spatially displaced, the range can be extended up to 1 km at direct line-of-sight per hop due to the limitations in the physical layer as well as the modulation used in Sub-1 GHz band to comply with IEEE 802.15.4 standard. Furthermore, when there are multiple nodes that are performing precision control for assembly, where the leader is not involved, the concept of *Decentralized Brains* is leveraged using the cluster-differentiate setting. A new *brain* is initialized using the 6LoWPAN network among the nodes that need to perform control in precise manner, thus decreasing the number of hops and also reducing the latency while propagating the states to the participating nodes in <3 ms. Cluster-differentiate setting is designed to handle up to 16 different leaders where the nodes can choose to participate in data replication. The rate at which data is replicated can be set up by the leader node, allowing for use case dependent frequency in data replication (i.e., the whole system is updated with its slow-changing state of assembly less-frequently than the updates required for quick, precise control between the interacting entities). This provides a guarantee in communication, which is critical for precise control, as well as modular communication such as point-to-point. Another use case of a cluster-differentiate setting is to selectively pick which nodes participate in the synchronous broadcast depending on their location to propagate the message for a spatially focused information penetration. For TESSERAE, a combination of RF (far) and LIDAR (near) ranging will be used for location determination, enabling tile proximity to be the trigger for activation of the multi-hop replication between tiles.

The border router with IPv6 adds the feasibility for communicating, to entities outside the network, trigger events such as *end of successful assembly* to a remote station. In the case of a sparse antenna array in space, on top of data replication, the accuracy in time synchronization can be used to trigger coherent beam forming. Given a 3 ms time delay for replication across nodes, we propose that this communication architecture could be used to share multi-part swarm position information throughout the network in an efficient manner [55]. Further study is necessary to determine whether this delay is low enough to support dynamically updating the phase shift required for coherent beam forming, in real time, with changing element positions (assuming a comparatively slow relative position change between

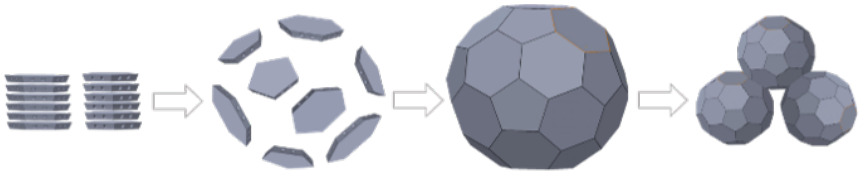

Fig. 5.2 Shipped Flat→Free Floating→ TESSERAE Assembly→Mosaic Constellation of multiple TESSERAE [105, 106]

elements of the array) [55]. The accuracy of time synchronization can be scaled to higher resolutions by increasing the operating clock frequency [61] with an accompanying energy consumption strain on the order of few mA. Note that stability in spacecraft assembly control will impose strict limits on allowable propagation delays that the network will need to accommodate.

5.2 Distributed Robotics

In the context of distributed robotics, *DezCom* – a decentralized context broker is proposed as a solution for the decentralization of logistics robots. Each robot is a node that communicates with a central server, a fleet manager in the current approach. The fleet manager maintains the context and communicates with each robot to facilitate coordination in the field. The fleet manager performs task planning as a primary task, but it is not limited to this function. It also performs other functions of coordination in context to multi-robot systems. In multi-robot systems, each robot performs tasks, where each task is allocated by the fleet manager centrally, and the robot performs the tasks independent of the fleet. There is no inter-communication enabled between the robots even though technology provides the possibility for communication. Using a decentralized context broker, the fleet manager is still a service in the network coordinating the task allocation. Nevertheless, the intelligence of motion planning and traffic planning can be abstracted to the robots. As proposed, the context broker allows for seamless communication among all the nodes in the network. When the fleet manager is not communicating with the robots, the robots can still organize themselves as a fleet to execute the pending tasks allocated by the fleet manager. The central dependency of an active fleet manager controlling the robots is reduced by decentralizing the communication where the robots can communicate and organize themselves. The fleet manager becomes a service in the network which allocates tasks and plans supplementary process logic for the robots. The robots can work with or without the fleet manager as the fleet's context is

communicated using the decentralized context broker. The history of the messages and the context of the processes persist in the broker's mempool, which allows the robots to query for required information about the process. When there are other integrative services for process optimization that need to be implemented, the integration becomes modular. Modularity is achieved due to the standard message bus interface provided by the decentralized context broker for the optimization service to query for the necessary information. Decentralization of services increases the communication overhead; therefore, it is necessary to choose the design guidelines where the decentralized context broker's features can be leveraged. With a centralized context broker such as MQTT, hyper scaling will overwhelm the communication resources available for service as nodes in the network require MQTT to execute tasks. With the decentralized context broker, the communication resources are localized. Every new node contributes to scaling and the robustness of the network with properties such as a distributed ledger, decentralized consensus, self-healing, and self-organization of the nodes. When a critical service such as a fleet manager is not available, the context broker provides the nodes with the certainty that the fleet manager is unavailable, which allows for deterministic monitoring and maintenance of the network. The design choices and requirements for a decentralized network in terms of high-power, high data-rate systems are discussed in Sec. 5.4

5.3 IoT Applications for Logistics

At an end to end IoT application, the decentralization of systems requires low-power low data-rate systems and high data-rate systems. The two types of systems are for the end nodes or low-power devices deployed in the field for reporting or monitoring physical parameters. The edge nodes or the cloud servers that run diverse business logic require coordination using a context broker. Here, the culmination of both the proposed solutions can be seen. Moreover, the requirements may not require all of the proposed solutions. The Sensor Floor as an application itself requires such a hybrid solution where 23 nodes collect information from 15 nodes each. In this case, a data aggregation platform is required where the data needs to be stored in chronological order as acquired from the low-power nodes.

In some instances, the end nodes do not require coordination with the other end nodes, where decentralization only incurs communication overhead. In most logistics use cases, the context broker provides the required decentralization in the edge network. The access points and edge routers are deployed out in the open. A trust-less network where any node

can join, organize, and communicate reduces the overhead of securing the assets and increases efficiency and scaling.

5.4 Design Guidelines

The minimum requirement of a decentralized network depends on the number of nodes and the scope of scalability. The scenario of the business logic and the dynamics of the system is the deciding factor for deploying decentralized networks in low-power, low data-rate systems. In high-power high data-rate systems, the proposed solutions' modularity allows for upgrading the communication layer to a decentralized network. Therefore, if the application logic allows for decentralization, then the number of participating nodes becomes the deciding factor for decentralized networking. Decentralization is vital in a highly dynamic system where there are multi-vendors and heterogeneous participants in the network. Trust and security are a concern for maintaining hyper-scalable industrial networks, but the organization and information transparency are critical factors that require decentralization. When it comes to decentralizing applications and systems communication in an open network or a trustless network where multiple parties participate in a network, blockchain-based applications prove to be the best fit. To apply blockchain as part of a decentralized application, one can easily follow the flowchart, as shown in Fig. 5.3 to understand if a subset of the features provided by blockchain-based solutions fit well for the requirements of the application to be developed.

In low data-rate systems, the necessity of decentralization might be required, but the system constraints might not allow for a truly decentralized application development approach. In such cases, edge routers can be deployed, which provide the necessary abstraction for decentralization, as seen in the centralized energy-neutral IoT testbed PhyNetLab. In this application, the nodes are centralized and communicate to one router. Still, these routers agree with a highly available, time-synchronized network that allows the nodes to be highly mobile and use the same amount of energy for communicating to its nearest edge router. During the design phase, the edge routers were implemented with a Hazelcast in-memory database, which allowed for synchronizing all the data structures necessary for the nodes' operation. These hybrid solutions are most viable in many cases as a truly decentralized system may be the right candidate for the application development approach—still, the system constraints introduce challenges such as memory and energy-intensive applications.

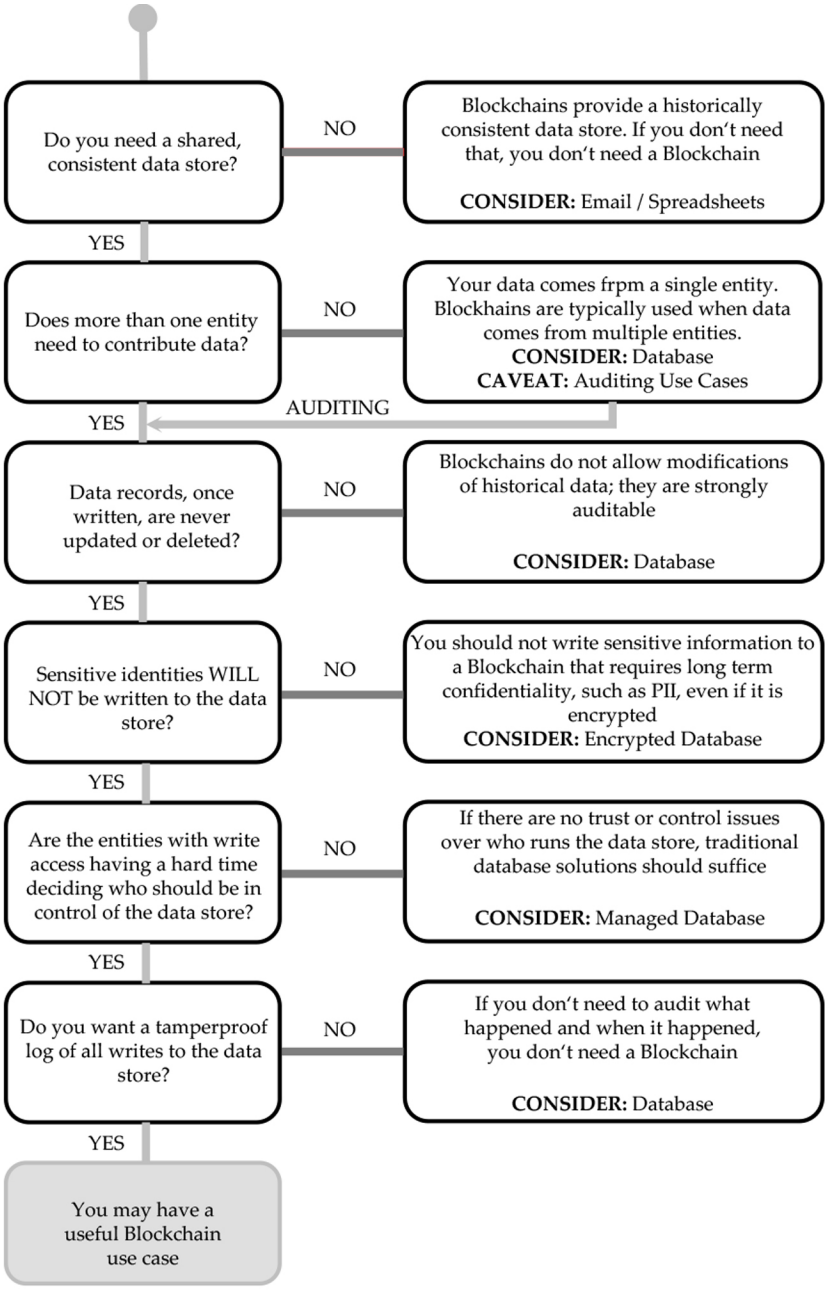

Fig. 5.3 A guide to application of blockchain as a solution to a problem [94].

Chapter 6
Conclusion

In this research, a holistic approach for developing a decentralized communication in Industry 4.0 systems has been proposed, developed, and deployed in industrial use cases. This chapter has three-sections, where Sec. 6.1 summarizes the proposal and the development in terms of the two system types. The outlook of the developed systems and the industrial viability of the developed solutions are discussed in Sec. 6.2. In the final Sec. 6.3, essential questions to pursue for the future and research notes for further extension of the developed solutions are discussed.

6.1 Summary

Decentralization of systems communication was the goal of this research, which gave rise to systems classification depending on communication constraints and available energy. The two systems were low-power, low data-rate systems, and high-power, high data-rate systems for which two solutions were proposed, developed, and evaluated for each of the systems. For the development of the Decentralized Brains, a network flooding primitive was developed to use the shared medium efficiently. It gave rise to a new method of deploying decentralized networks in low-power low data-rate systems. Several use cases required such a communication paradigm for collaborative sensing and self-assembly maneuvers. The proposed solution is developed from a logistics system's perspective and requirements, but its features are not limited only to logistics. In Sec. 5.1, the application of the concept of *Decentralized Brains* was explored, and specific concepts were developed, deployed, and tested in zero-gravity experiments. The communication primitive was tested with 345 nodes in the Sensor Floor, for developing the experiments, an open-source software framework was chosen called the Contiki-OS. The developed communication primitives are

developed as extensions for the Contiki-OS to choose the necessary communication primitives depending on the scenario selectively. The codebase for the developed Contiki-OS extensions are made available in the code repository available at `https://www.github.com/akrv/deBr`

The industrial data networks with high-power, high data-rate systems also require decentralization for robust functioning. Decentralization also helps to reduce bottlenecks due to system unavailability or network unavailability. The developed solution called the DezCom is a decentralized context broker. The context broker functions as a middleware between the communicating systems, providing consensus for the sent data, and providing a secure communication bus where trust-less systems can also participate in collaborative actions in an industrial scenario. A blockchain-based system is designed and developed for testing with a multi-robot industrial environment. Various approaches were explored to identify the prospective candidates for the development of the DezCom context broker. The system was deployed globally in data centers, and the data was published and subscribed to within the research facility to demonstrate the global availability of the developed solution. The proposed solution outperforms the most commonly used context broker in some instances where MQTT reaches its operational limits due to its design. However, in some cases, due to the simplicity of the MQTT's design, it outperforms the proposed solutions. These cases are not as critical as the developed system is targeted for hyper-scalable, mMTC, which is a requirement for developing Industry 4.0 applications. Therefore, in chapter 5, various use cases for the *Decentralized Brains* and DezCom context broker are discussed, finally summarizing the design guidelines for deploying a decentralized communication network.

6.2 Outlook

In chapter 5, the two developed systems were deployed in diverse industrial applications, and the design guidelines are proposed for decentralization of the Industry 4.0 communication. It is evident that with hyper scaling of IoT devices in the field and the trend for mMTC, decentralization is necessary. Decentralization means shifting the power of communication, computation, and intelligence from the center to the network's edges. The network can communicate for reacting to events without the oversight of a central management system. The systems perform any decision other than the process or business logic in the edge and the field. In contrast, decision-making capabilities are abstracted as a part of the system existing in the same hierarchy as other nodes. This allows

for hyper scalability, where the tasks are performed with the available intelligence at the edges. When there are control effects that need to be implemented, these nodes can actively change the parameters to allow for such changes in the run time. This decouples the requirement of a centralized, always available infrastructure for a full functioning network. Decentralization of communication allows for decentralizing the whole network as most of the distributed systems rely heavily on other systems' availability. In such an architecture, failures are mitigated by allowing each system to function autonomously with its local intelligence.

6.3 Discussion and Future

Decentralized Brains is developed as an extension of an open-source embedded real-time operating system called the Contiki-OS with the goal that it is one of the mature frameworks for developing IoT applications. The developed communication primitives are modular and are developed with the perspective of platform-agnostic development. The time synchronization and other specific libraries used concerning the target hardware can be abstracted in the future for using other alternatives when there are hardware limitations. This allows for easy industrial adaptation for time-synchronized, low-power low data-rate systems. Moreover, the developed system is a reference implementation for demonstrating the capabilities that can be achieved for developing methods that are akin to computers due to their fast computational response and wide availability of sensing methods for interacting with the communication medium.

The development of DezCom demonstrates a context broker that can provide a secure layer for communication abstracting the decentralized byzantine fault-tolerant consensus and providing a standard web-socket interface for developing industrial applications. Demonstration of DezCom also provides other existing context brokers with a proof to develop decentralized context brokers with application layer protocol for payload validation and replication of data structures as well as files. Node discovery is available as part of the communication layer for associating nodes into the network. Future development should consider developing application layer protocols for service discovery and other application-specific protocols for process control using the system to develop industrial applications. The NGSI standardization further simplifies the application development process, which can be integrated into the decentralized context broker to deploy the software into existing industrial applications directly.

References

[1] J. Macaulay, L. Buckalew, and G. Chung, "Internet of things in logistics," *DHL Trend Research*, vol. 1, no. 1, pp. 1–27, 2015.

[2] K. M. Dudzinski, J. A. Thomas, and J. D. Gregg, "Communication in marine mammals," in *Encyclopedia of marine mammals*, Elsevier, 2009, pp. 260–269.

[3] P. Godfrey-Smith, *Other minds: The octopus, the sea, and the deep origins of consciousness*. Farrar, Straus and Giroux, 2016.

[4] E. Musk *et al.*, "An integrated brain-machine interface platform with thousands of channels," *Journal of medical Internet research*, vol. 21, no. 10, e16194, 2019.

[5] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, 2018. DOI: 10.1109/MM.2018.032271057.

[6] M. M. Abdelaziz and J. C. Soto, *Method and apparatus for decentralized device and service description and discovery*, US Patent 7,656,822, 2010.

[7] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *USENIX Annual Technical Conference*, 2014, pp. 305–319.

[8] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[9] A. K. Ramachandran Venkatapathy, H. Bayhan, F. Zeidler, and M. ten Hompel, "Human machine synergies in intra-logistics: Creating a hybrid network for research and technologies," in *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on*, IEEE, 2017, pp. 1065–1068.

[10] A. K. R. Venkatapathy, A. Riesner, M. Roidl, J. Emmerich, and M. ten Hompel, "Phynode: An intelligent, cyber-physical system with energy neutral operation for phynetlab," in *Smart SysTech 2015; European Conference on Smart Objects, Systems and Technologies; Proceedings of*, VDE, 2015, pp. 1–8.

[11] A. K. R. Venkatapathy, M. Roidl, A. Riesner, J. Emmerich, and M. ten Hompel, "Phynetlab: Architecture design of ultra-low power wireless sensor network testbed," in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE, 2015, pp. 1–6.

[12] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[13] A. Willig, "Recent and emerging topics in wireless industrial communications: A selection," *Industrial Informatics, IEEE Transactions on*, vol. 4, no. 2, pp. 102–124, 2008.

[14] M. Roidl, J. Emmerich, A. Riesner, M. Masoudinejad, D. Kaulbars, C. Ide, C. Wietfeld, and M. Ten Hompel, "Performance availability evaluation of smart devices in materials handling systems," in *2014 IEEE/CIC International Conference on Communications in China-Workshops (CIC/ICCC)*, IEEE, 2014, pp. 6–10.

[15] A. Pal and K. Kant, "Water flow driven sensor networks for leakage and contamination monitoring," in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE, 2015, pp. 1–9.

[16] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*. John Wiley & Sons, 2010, vol. 4.

[17] K. Römer and F. Mattern, "The design space of wireless sensor networks," *Wireless Communications, IEEE*, vol. 11, no. 6, pp. 54–61, 2004.

[18] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, IEEE, 2005, pp. 364–369.

[19] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A low-cost, 3d wireless sensor network testbed," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, Springer, 2012, pp. 302–316.

[20] S. H. Lee, S. Lee, H. Song, and H. S. Lee, "Wireless sensor network design for tactical military applications: Remote large-scale environments," in *Military Communications Conference, 2009. MILCOM 2009. IEEE*, IEEE, 2009, pp. 1–7.

[21] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM Sigplan notices*, vol. 35, pp. 93–104, 2000.

[22] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, ACM, 1999, pp. 263–270.

[23] M. A. M. Vieira, C. N. Coelho Jr, D. da Silva, and J. M. da Mata, "Survey on wireless sensor network devices," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*, IEEE, vol. 1, 2003, pp. 537–544.

[24] D. Chen and P. K. Varshney, "Qos support in wireless sensor networks: A survey.," in *International Conference on Wireless Networks*, vol. 13244, 2004, pp. 227–233.

[25] A. J. Goldsmith and S. B. Wicker, "Design challenges for energy-constrained ad hoc wireless networks," *Wireless Communications, IEEE*, vol. 9, no. 4, pp. 8–27, 2002.

[26] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *CSE Journal Articles*, 2006.

[27] J. A. Gutierrez, E. H. Callaway, and R. L. Barrett, *Low-rate wireless personal area networks: enabling wireless sensors with IEEE 802.15. 4*. IEEE Standards Association, 2004.

[28] Texas Instruments, "CC1200 Low-Power, High-Performance RF Transceiver", SWRS123D Datasheet, July 2013 [Revised OCT. 2014].

[29] Texas Instruments,"System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications", SWRS081B Datasheet, April 2009 [Revised Feb. 2011].

[30] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi," in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, IEEE, 2007, pp. 46–51.

[31] J. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, B. Heile, *et al.*, "Ieee 802.15. 4: A developing standard for low-power low-cost wireless personal area networks," *network, IEEE*, vol. 15, no. 5, pp. 12–19, 2001.

[32] W. Stallings, *Networking standards: a guide to OSI, ISDN, LAN, and MAN standards*. Addison-Wesley Longman Publishing Co., Inc., 1993.

[33] J. Elson and K. Römer, "Wireless sensor networks: A new regime for time synchronization," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 149–154, 2003.

[34] K. Römer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks," *Handbook of sensor networks: Algorithms and architectures*, vol. 49, p. 199, 2005.

[35] K. Correll, N. Barendt, and M. Branicky, "Design considerations for software only implementations of the ieee 1588 precision time protocol," in *Conference on IEEE*, vol. 1588, 2005, pp. 11–15.

[36] G. Mace, A. Van Kempen, S. Kreuzer, and G. Neville-Neil, *Precision time protocol daemon*, `http://www.manualpages.de/FreeBSD/FreeBSD-ports-9.0-RELEASE/man8/ptpd2.8.html`.

[37] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, "How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective," *International Journal of Science, Engineering and Technology 8 (1), 37*, vol. 44, 2014.

[38] Texas Instruments, "2MHz, 600mA Step-Down DC-DC Converter", lm3671 SNVS294R ,Datasheet, November 2004 [Revised Nov. 2013].

[39] Texas Instruments,"Cost-Effective Voltage and Current Protection Integrated Circuit for Single-Cell Li-Ion/Li-Polymer Batteries", BQ29700 SLUSBU9A Datasheet, March 2014 [Revised June 2014].

[40] Texas Instruments, "CC120X Low-Power High Performance Sub-1 GHz RF Transceivers", SWRU346B User Guide, July 2013 [Revised OCT. 2014].

[41] M. Masoudinejad, J. Emmerich, D. Kossmann, A. Riesner, M. Roidl, and M. ten Hompel, "Development of a measurement platform for indoor photovoltaic energy harvesting in materials handling applications," in *Renewable Energy Congress (IREC), 2015 6th International*, IEEE, 2015, pp. 1–6.

[42] Maxim Integrated, "Industry's Lowest-Power Ambient Light Sensor with ADC", MAX44009 Datasheet, January 2011 [Not Revised].

[43] Maxim Integrated, "RGB Color, Infrared, and Temperature Sensors", MAX44006-MAX44008 Datasheet, July 2012 [Revised Aug. 2012].

[44] Texas Instruments,"Ultra Low Power Boost Charger with Battery Management and Autonomous Power Multiplexor for Primary Battery in Energy Harvester Applications", bq25505 SLUSBJ3D Datasheet, August 2013 [Revised Jan. 2014].

[45] W. Gay, *Raspberry Pi hardware reference*. Apress, 2014.

[46] Texas Instruments, "A USB-Enabled System-On-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications", SWRS086A Datasheet, September 2009 [Revised JUN. 2010].

[47] B. Richardson, K. Leydon, M. Fernstrom, and J. A. Paradiso, "Z-tiles: Building blocks for modular, pressure-sensing floorspaces," in *CHI'04 extended abstracts on Human factors in computing systems*, 2004, pp. 1529–1532.

[48] J. A. Paradiso, K.-y. Hsiao, J. Strickon, J. Lifton, and A. Adler, "Sensor systems for interactive surfaces," *IBM Systems Journal*, vol. 39, no. 3.4, pp. 892–914, 2000.

[49] J. Paradiso, C. Abler, K.-y. Hsiao, and M. Reynolds, "The magic carpet: Physical sensing for immersive environments," in *CHI'97 Extended Abstracts on Human Factors in Computing Systems*, Association for Computing Machinery, 1997, pp. 277–278.

[50] R. Falkenberg, M. Masoudinejad, M. Buschhoff, A. K. R. Venkatapathy, D. Friesel, M. ten Hompel, O. Spinczyk, and C. Wietfeld, "Phynetlab: An iot-based warehouse testbed," in *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on*, IEEE, 2017, pp. 1051–1055.

[51] T. Instruments, "Cc1350 datasheet: Cc1350 soc with rf core," *URL: http://www.ti.com/lit/ds/symlink/cc1350.pdf*, 2016.

[52] *CC1350 sensor tag design files*, en. [Online]. Available: `http://www.ti.com/lit/zip/swrc321` (visited on 03/26/2020).

[53] E. Wollert, "CC2538/CC26x0/CC26x2 Serial Bootloader Interface," en, *Texas Instruments*, p. 21, 2015.

[54] Y. Rogers and J. Ellis, "Distributed cognition: An alternative framework for analysing and explaining collaborative working," *Journal of information technology*, vol. 9, no. 2, pp. 119–128, 1994.

[55] A. K. R. Venkatapathy, A. Ekblaw, M. ten Hompel, and J. Paradiso, "Decentralized brain in low data-rate, low power networks for collaborative manoeuvres in space," in *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, IEEE, 2018, pp. 83–88.

[56] A. K. R. Venkatapathy and M. ten Hompel, "A decentralized context broker using byzantine fault tolerant consensus," *Ledger*, vol. 4, 2019.

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

[58] Z. Alliance, "Zigbee specification. 2012," *ZigBee Document 053474r20*, 2012.

[59] B. SIG, "Specification of the bluetooth system core package version 4.2," *Bluetooth Document: Master Table of Contents and Compliance Requirements*, 2014.

[60] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, Apr. 2011, pp. 73–84.

[61] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010, pp. 151–161.

[62] T. Instruments, "Cc430 datasheet: Msp430 soc with rf core," *URL: http://www.ti.com/lit/ds/symlink/cc430f5137.pdf*, 2013.

[63] M. Zimmerling, L. Mottola, and S. Santini, "Synchronous transmissions in low-power wireless: A survey of communication protocols and network services," *arXiv preprint arXiv:2001.08557*, 2020.

[64] A. Swami, Q. Zhao, Y.-W. Hong, and L. Tong, *Wireless sensor networks: signal processing and communications perspectives*. John Wiley & Sons, 2007.

[65] J. Lu and K. Whitehouse, *Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks*. IEEE, 2009.

[66] K. Leentvaar and J. Flint, "The capture effect in fm receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 531–539, 1976.

[67] D. Davis and S. Gronemeyer, "Performance of slotted aloha random access with delay capture and randomized time of arrival," *IEEE Transactions on Communications*, vol. 28, no. 5, pp. 703–710, 1980.

[68] C.-H. Liao, Y. Katsumata, M. Suzuki, and H. Morikawa, "Revisiting the so-called constructive interference in concurrent transmission," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, IEEE, 2016, pp. 280–288.

[69] M. Doddavenkatappa, M. C. Chan, B. Leong, *et al.*, "Splash: Fast data dissemination with constructive interference in wireless sensor networks.," in *NSDI*, 2013, pp. 269–282.

[70] B. A. Nahas, S. Duquennoy, and O. Landsiedel, "Network-wide Consensus in Low-Power Wireless Networks," in *SenSys: Proceedings of the 15th ACM Conference on Embedded Networked Sensor Systems*, Nov. 2017.

[71] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[72] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems: concepts and design*. pearson education, 2005.

[73] P. Muri and J. McNair, "A survey of communication sub-systems for intersatellite linked systems and cubesat missions.," *JCM*, vol. 7, no. 4, pp. 290–308, 2012.

[74] T. Winter, P. Thubert, A. Brandt, J Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R Alexander, "Rpl: Ipv6 routing protocol for low-power and lossy networks," Internet Engineering Task Force (IETF), Tech. Rep., 2012.

[75] J. Dregger, J. Niehaus, P. Ittermann, H. Hirsch-Kreinsen, and M. ten Hompel, "The digitization of manufacturing and its societal challenges: A framework for the future of industrial labor," in *Ethics in Engineering, Science and Technology (ETHICS), 2016 IEEE International Symposium on*, IEEE, 2016, pp. 1–3.

[76] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 541–549.

[77] A. Cameron, M. Payne, and B. Prela, "Research and implementation of multiple blockchain byzantine secure consensus protocols for robot swarms," *http://agnescameron.info*, 2018.

[78] H. Chen, T. Finin, A. Joshi, *et al.*, "A context broker for building smart meeting rooms," in *Proceedings of the AAAI Symposium on Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium*, 2004, pp. 53–60.

[79] A. K. Dey and G. D. Abowd, "Providing architectural support for building context-aware applications," *PhD thesis*, 2000.

[80] M. Lamming and M. Flynn, "Forget-me-not: Intimate computing in support of human memory," in *Proc. FRIEND21, 1994 Int. Symp. on Next Generation Human Interface*, 1994, p. 4.

[81] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 32–43.

[82] T. Kindberg and J. Barton, "A web-based nomadic computing system," *Computer Networks*, vol. 35, no. 4, pp. 443–456, 2001.

[83] J. Lin, R. Laddaga, and H. Naito, "Personal location agent for communicating entities (place)," in *International Conference on Mobile Human-Computer Interaction*, Springer, 2002, pp. 45–59.

[84] M. H. Coen *et al.*, "Design principles for intelligent environments," in *AAAI/IAAI*, 1998, pp. 547–554.

[85] W. N. Schilit, "A system architecture for context-aware molbil computing," PhD thesis, Columbia University New York, NY, 1995.

[86] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 1, pp. 91–102, 1992.

[87] E. NGSI-LD, "Context information management (cim) and application programming interface (api)," *ETSI GS CIM*, vol. 4, p. V1, 2018.

[88] e. FIWARE Foundation, *Fiware: The open source platform for our smart digital https://www.fiware.org*, 2018. [Online]. Available: `https://www.fiware.org/`.

[89] P. Salhofer, "Evaluating the fiware platform," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[90] V. Araujo, K. Mitra, S. Saguna, and C. Åhlund, "Performance evaluation of fiware: A cloud-based iot platform for smart cities," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 250–261, 2019.

[91] A Banks, E Briggs, K Borgendale, and R Gupta, "Mqtt version 5.0," *OASIS Standard*, 2019.

[92] M. Handosa, D. Gračanin, and H. G. Elmongui, "Performance evaluation of mqtt-based internet of things systems," in *2017 Winter Simulation Conference (WSC)*, IEEE, 2017, pp. 4544–4545.

[93] L. Lamport, "The weak byzantine generals problem," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 668–676, 1983.

[94] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," *arXiv preprint arXiv:1906.11078*, 2019.

[95] J.-P. Bahsoun, R. Guerraoui, and A. Shoker, "Making bft protocols really adaptive," in *2015 IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2015, pp. 904–913.

[96] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*, Springer, 1987, pp. 369–378.

[97] A. Langley, E. Kasper, and B. Laurie, *Rfc 6962-certificate transparency*, 2013.

[98] S. Popov, "The tangle," *cit. on*, p. 131, 2016.

[99] ——, "Iota: Feeless and free," *IEEE Blockchain technical briefs*, 2019.

[100] S. Popov, O. Saa, and P. Finardi, "Equilibria in the tangle," *Computers & Industrial Engineering*, vol. 136, pp. 160–172, 2019.

[101] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," PhD thesis, The University of Guelph, 2016.

[102] ——, "Byzantine fault tolerant state machine replication in any programming language," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 546–546.

[103] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, 2014.

[104] C. Blesing, D. Luensch, J. Stenzel, and B. Korth, "Concept of a multi-agent based decentralized production system for the automotive industry," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, Springer, 2017, pp. 19–30.

[105] A. Ekblaw and J. Paradiso, "Self-assembling space structures: Buckminsterfullerene sensor nodes," in *AIAA/AHS Adaptive Structures Conference*, 2018, p. 0565.

[106] ——, "Tesserae: Self-assembling shell structures for space exploration," in *Proceedings of the Annual Symposium of the International Association of Shell and Spatial Structures: Extraplanetary Architecture*, 2018, p. 0317.

[107] I. Bekey, *Advanced space system concepts and technologies: 2010-2030+*. AIAA, 2003.

[108] F. L. Lewis *et al.*, "Wireless sensor networks," *Smart environments: technologies, protocols, and applications*, pp. 11–46, 2004.

[109] D. Hoover, "Design, testing, and implementation of wisemote: A wireless sensor network for structural health monitoring," PhD thesis, Texas Tech University, 2012.

[110] M. Corporation, *Telos ultra low power ieee 802.15.4 compliant wireless sensor module revision b : Humidity, light, and temperature sensors with usb*, Online, 2004.

[111] B. Cody-Kenny, D. Guerin, D. Ennis, R. Simon Carbajo, M. Huggard, and C. Mc Goldrick, "Performance evaluation of the 6lowpan protocol on micaz and telosb motes," in *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, ACM, 2009, pp. 25–30.

[112] J Hill, R Szewczyk, A Woo, and D Culler, "Tinyos," *University of California, Berkeley. URI: http://www.tinyos.net/. Cited*, vol. 10, no. 06, 2009.

[113] Texas Instruments, "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver", SWRS041c Datasheet, http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[114] Texas Instruments, "MSP430F471x3, MSP430F471x6, MSP430F471x7 MIXED SIGNAL MICROCONTROLLER", SLAS626C Datasheet, http://www.ti.com/lit/ds/symlink/msp430f47187.pdf.

[115] Digi, *Xbee® multipoint rf modules*, Online Datasheet https://www.sparkfun.com/Datasheets/Wireless/Zigbee/XBee-XCS-Datasheet.pdf.

[116] I. I. B. Sensor, *Crossbow technology inc*.

[117] U. of Illinois at Urbana Champaign, *Ism400 multimetric imote2 sensor board datasheet and user's guide*, Illinois Structural Health Monitoring Project, 2009.

[118] Texas Instruments, "MSP430FR5969 16 MHz Ultra-Low-Power Microcontroller featuring 64 KB FRAM, 2 KB SRAM, 40 IO", SLAS704E Datasheet, October 2012 [Revised Mar. 2015].

[119] Gartner, *The Internet of Things (IoT)* units installed base by category from 2013 to 2020 (in millions)*, http://www.statista.com/statistics/370350/internet-of-thingsinstalled-base-by-category/, [Online; accessed September 02, 2015], 2013.

[120] PAC, *Projected global revenue of the "Internet of Things" from 2007 to 2020 (in million euros)*, http://www.statista.com/statistics/270829/projected-global-revenue-ofthe-internet-of-things/, [Online; accessed September 02, 2015], 2013.

[121] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, IEEE Press, 2005, p. 68.

[122] D. Sakamuri and H. Zhang, "Elements of sensornet testbed design," *World Scientific Publishing Co*, pp. 1–36, 2009.

[123] R. Govindan *et al.*, *Tutornet: A tiered wireless sensor network testbed*.

[124] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat, "Mirage: A microeconomic resource allocation system for sensornet testbeds," in *The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II.*, IEEE, 2005, pp. 19–28.

[125] *Kamin whitehouse :: Projects :: Vinelab wireless testbed*, `http://www.cs.virginia.edu/~whitehouse/research/testbed/`, (Accessed on 11/01/2020).

[126] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal, "Kansei: A high-fidelity sensing testbed," *Internet Computing, IEEE*, vol. 10, no. 2, pp. 35–47, 2006.

[127] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, "Wisebed: An open large-scale wireless sensor network testbed," in *Sensor Applications, Experimentation, and Logistics*, Springer, 2010, pp. 68–87.

[128] T. Baumgartner, I. Chatzigiannakis, S. P. Fekete, S. Fischer, C. Koninis, A. Kröller, D. Krüger, G. Mylonas, and D. Pfisterer, "Distributed algorithm engineering for networks of tiny artifacts," *Computer Science Review*, vol. 5, no. 1, pp. 85–102, 2011.

[129] O. Hahm, M. Günes, and K. Schleiser, "Des-testbed a wireless multi-hop network testbed for future mobile networks," in *Proc. of GI/ITG KuVS Workshop on Future Internet (KuVS), Hannover, Germany*, 2010.

[130] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, "The w-ilab. t testbed," in *Testbeds and Research Infrastructures. Development of Networks and Communities*, Springer, 2011, pp. 145–154.

[131] C. B. Des Rosiers, G. Chelius, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, and T. Noël, "Senslab very large scale open wireless sensor network testbed," in *Proc. 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM)*, 2011.

[132] M. Sridharan, W. Zeng, W. Leal, X. Ju, R. Ramnath, H. Zhang, and A. Arora, "Kanseigenie: Software infrastructure for resource management and programmability of wireless sensor network fabrics," *Next Generation Internet Architectures and Protocols*, pp. 275–300, 2010.

[133] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "Twist: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, ACM, 2006, pp. 63–70.

[134] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A survey on facilities for experimental internet of things research," *Communications Magazine, IEEE*, vol. 49, no. 11, pp. 58–67, 2011.

[135] D. Schwerdel, D. Hock, D. Günther, B. Reuther, P. Müller, and P. Tran-Gia, "Tomato-a network experimentation tool," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, Springer, 2012, pp. 1–10.

[136] B. Hullár, S. Laki, J. Stéger, I. Csabai, and G. Vattay, "Sonoma: A service oriented network measurement architecture," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, Springer, 2012, pp. 27–42.

[137] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, *et al.*, "Flexible experimentation in wireless sensor networks," *Communications of the ACM*, vol. 55, no. 1, pp. 82–90, 2012.

[138] M. Güneş, B. Blywis, F. Juraschek, and P. Schmidt, "Practical issues of implementing a hybrid multi-nic wireless mesh-network," *Technical report*, 2008. DOI: 10.17169/REFUBIUM-22638. [Online]. Available: https://refubium.fu-berlin.de/handle/fub188/18961.

[139] T. Baumgartner, I. Chatzigiannakis, S. Fekete, C. Koninis, A. Kröller, and A. Pyrgelis, "Wiselib: A generic algorithm library for heterogeneous sensor networks," in *Wireless Sensor Networks*, Springer, 2010, pp. 162–177.

[140] A. G. TU Braunschweig IBR, *Wiselib documentation*, https://github.com/ibr-alg/wiselib/wiki.

[141] O. Fambon, E Fleury, G. Harter, R. Pissard-Gibollet, and F. Saint-Marcel, "Fit iot-lab tutorial: Hands-on practice with a very large scale testbed tool for the internet of things," *10èmes journées francophones Mobilité et Ubiquité, UbiMob2014*, 2014.

List of Abbreviations

2-PC	2 Phase commit
3-PC	3 Phase commit
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ABCI	Application Blockchain Interface
AGV	Autonomous Ground / Guided Vehicle
BFT	Byzantine Fault Tolerance
BLE	Bluetooth Low Energy
BPSK	Binary Phase Shift Keying
CCA	Clear Channel Assessment
Contiki-OS	Contiki-Open Source Operating System
CPS	Cyber Physical Systems
CSMA	Carrier Sense Multiple Access
CubeSats	U-class spacecrafts
DAG	Directed Acyclic Graph
DODAG	Destination Oriented Directed Acyclic Graph
DSSS	Direct-sequence Spread Spectrum
FMS	Fleet Management System
FSK	Frequency Shift Keying
GFSK	Gaussian Frequency Shift Keying

GPIO	General Purpose Input Output
IMU	Inertial Measurement Unit
IoT	Internet of Things
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
KPI	Key Performance Indicator
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Networks
MAC	Medium Access Control
MCU	Micro-Controller Unit
mMTC	Massive Machine Type Communications
MNB	Master Network Board
MQTT	Message Queuing Telemetry Transport
MTU	Maximum Transmission Unit
NGSI-LD API	API for Context Information Management
O-QPSK	Offset Quadrature Phase Shift Keying
OOK	On-Off Keying
P2P	Peer-2-Peer
PCB	Printed Circuit Board
RF	Radio Frequency
RPi	Raspberry-Pi Single Board Computer
RSSI	Received Signal Strength Indicator
SFD	Start of Frame Delimiter
SMARTFACE	Smart Micro Factory for Electric Vehicles with Lean Production Planning
SoC	System on Chip
SPI	Serial Peripheral Interface
SSB	Swappable Secondary Board
SSID	Service Set Identifier

TBTT	Target Beacon Transmission Time
TESSERAE	Tessellated Electromagnetic Space Structures for the Exploration of Re-configurable, Adaptive Environments
TPU	Tensor Processing Unit
UART	Universal Synchronous Asynchronous Receiver Transmitter
UUID	Universal Unique Identifier
VHT	Virtual High-resolution Time
WMS	Warehouse Management System
WSN	Wireless Sensor Networks

Appendix A
Appendix

All's well that ends well

A.0.1 Wireless sensor nodes

In this section, various WSN field nodes are compared to provide a benchmark for a research hardware platform. PhyNode is a hardware platform developed for a PhyNetLab testbed. It reveals the design specification and provides the comparable features with existing WSN nodes in WSN testbeds. The survey of the various WSN nodes provides the design specification and the feature set that is to be implemented in PhyNode. The different types of wireless sensor nodes are discussed followed by the detailed description of existing WSN field nodes. The three types of nodes ordered with respect to their responsibility in the network:

1. Controller nodes
2. Router nodes
3. End nodes

A.0.2 Types of nodes in WSN

In case of a wireless network, a controller node in the network controls the medium of communication creating defined slots for transmission and giving access to the medium for other nodes to successfully transmit. The controller node also provides identification services such as providing token-based address with timeout or physical address of the nodes matched to the application address. Additionally, they can collect all the data generated in the network and also translate inbound and outbound traffic from heterogeneous networks.

Router nodes perform their function to monitor and report. Additionally, they also forward traffic from the nodes that are not in the coverage of the

controller nodes. They bind the end nodes with the controller nodes. Router nodes are the key in a multi-hop network where the data packet has to be forwarded through other nodes to reach the destination. Router nodes also maintain a routing table either of the next neighbor nodes or the whole map of the network, depending on the type of routing algorithm. End nodes are energy-constrained nodes deployed in the field and they are not monitored for a longer period of time [108]. In an unstructured WSN, they often are out of coverage due to a change in the host environment or due to the mobility of the nodes. They opportunistically transmit data from a queue when in range with a data sink or a neighboring node in a mesh network. They are called reduced function devices. These nodes only perform the core responsibility of the network, which is to transmit the packet it generates and to respond to the protocol based messages from the router or controller nodes.

There is a sink or a base station which communicates with these sensor nodes and collects the data. The sink node also transmits information from the user that is used for actuating the physical entity on which the wireless sensor node is attached. It is combination of a router node and a controller node. A base station can be router node but a router is not a base station. The user uses the data generated from the sensor nodes to perform analysis on different levels such as facility monitoring, inventory management etc. This sink node or the base station is usually connected to the Internet where the data is stored, monitored and analyzed. In some cases the sink node does not have internet connection due to the spatial location but has a large memory attached to it where data is written for later analysis [109]. These systems can either utilize resources on demand or are always available depending on the application. The communication architecture for WSNs would be to connect all the sensor nodes in the field to one or more base stations. Depending on the infrastructure it can be decided for the number of nodes to be implemented in the facility with the number of base stations [12]. At times it is possible that the sensor node does not have a direct connection to the base station due to the radio link or the range of coverage. This node is called an isolated node. At such conditions protocol standards with self-healing, intelligent routing schemes should be used where the data packets reach the base station through the neighboring nodes from the sensor node.

As described in the section A.0.2 about the different types of sensor nodes that can be deployed in a WSN, this neighbor node acts as a router node. In the above-mentioned scenario the router nodes that are connected to the isolated node forward the packets to the base station. This not only requires a specially designed routing algorithm but these nodes should also have enough power so that they can listen to packets from the isolated nodes and also performs its functions as a sensor node. Network topology plays an important role in designing the type of nodes that have to be placed into the implementation.

A.0.3 Existing WSN field nodes

Wireless sensor network field nodes are built with respect to their application. There is a trade off between the features and the application requirements. In this section existing field nodes, WiSeMote and Telosb, are studied and their features are listed with the features of PhyNode. These two wireless sensor nodes are specifically selected, since they also have Texas Instruments manufactured MSP430 series MCU units as the processing core of the nodes. The system design of PhyNode is described with the description of the implementation of the listed features [10].

Definition of Mote

The meaning of mote, as defined by the Oxford dictionary, is a small particle; a speck. This term is used to mark a sensor node as it is a small particle in the environment. Another definition for usage in technology defined by researchers from the two research centres Network Embedded Systems Technology (NEST) Berkeley and Center for Embedded Networked Sensing (CENS) for "mote" is it can be a tiny computer with networking capabilities for remote sensing.

A.0.3.1 Telosb

Telosb is an ultra-low power wireless module that is used in sensor networks, for monitoring applications, and rapid application prototyping, which was initially designed and published by UC Berkeley [110]. Telosb is implemented with industry standards like USB and IEEE 802.15.4. With these standards interoperability seamlessly out-ofthe- box with other devices can be implemented with other devices that implement these standards. Sensors for humidity, temperature, light and providing flexible interconnection with peripherals, are integrated with the sensor node. Revision B of the Telos sensor node is called Telosb and includes increased performance, functionality, and expansion, using the sensor suite compared to the previous version [111]. Telosb supports TinyOS out-of-the-box which was developed at the same research facility [112]. TinyOS is a set of cooperating processes and tasks written in the nesC programming language [112]. It provides mesh networking of wireless sensor nodes. It is also compatible with Contiki, an open operating system for WSNs [57]. Salient features of the CC2420 are that it works on 2.4 GHz band with IEEE 802.15.4 standard, it is ZigBee ready RF frontend with low power and low voltage consumption with no requirement for external RX filter or switch

[113]. The maximum data-rate provided by this standard is 250 kbit/s, which is also the maximum data-rate provided by Telosb. The transceiver has a 256 byte FIFO (first in first out) register that is divided into two equal halves and used for receiving and transmitting a data buffer, which can be accessed via a serial peripheral interface.

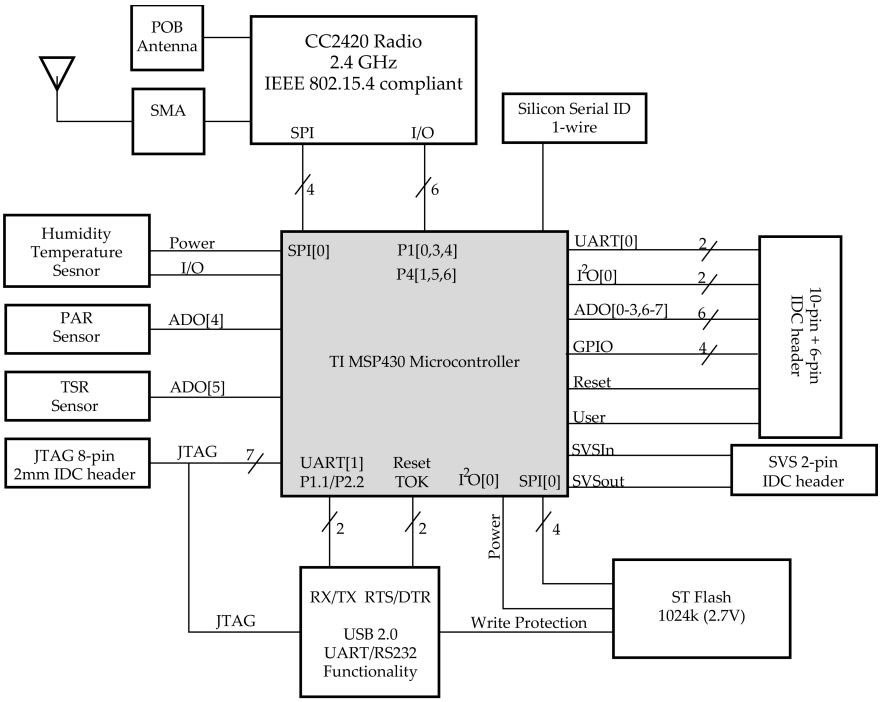

Fig. A.1 Functional block diagram of the Telosb module as illustrated in datasheet [110]

Figure A.1 shows the functional block diagram of the Telosb module. The processing unit of Telosb has an 8 MHz (operating frequency) Texas Instruments MSP430 MCU as the central processing unit with a 10k Random Access Memory (RAM) and a 48k flash memory storage [110]. It has an integrated ADC, DAC, Supply Voltage Supervisor, and Direct Memory Access (DMA) Controller [110]. The CC2420 supports an integrated onboard antenna with a 50m range indoors and 125m range outdoors, even though this highly depends on the path loss in the environment and the possibility of direct line of sight for the radio frequency between the transmitting and receiving stations [110]. The Telosb mote has a USB programming interface and data collection can be enabled through the onboard USB controller. The feature set can be extended using the 16-pin expansion that is interfaced with the MCU unit [110]. An optional

SMA antenna connector can be added to improve the antenna gain in case the PCB antenna proves to be inefficient [110].

Two AA batteries can power Telosb with anhe operating voltage in the range of 2.1 to 3.6 V DC. The voltage provided by the AA batteries must be at least 2.7 V when programming the micro-controller flash or external flash [110]. The Telosb can be powered also using the USB when connected with a host computer. The operating voltage is at 3 V when it is connected with the USB. Size of the module fits exactly two AA batteries by design but the battery pack is not necessary when the mote is connected using USB [110]. Another alternative power input is via the 16-pin expansion to the mote. At all times, the mote should not be powered with more than 3.6 V, which is damages the internal components [110].

A.0.3.2 WiSeMote

WiSeMote system [109] is a WSN node, developed in context to measuring and monitoring the structural health of buildings. WiSeMote is a fully integrated ultra-low power wireless smart sensor node that is used in structural health monitoring with the hardware design focused towards implementing unattended networks with long term monitoring applications. The faculty of civil engineering at the Texas Tech University developed it. WiSeMote system is a wireless networked system with low data-rate operating at IEEE 802.15.4 standard in 2.4 GHz band. It was developed with a base station that acts as a data sink and operates the low-power WSN nodes. It was developed to provide improvement in the network reliability, measurement noise mitigation, power consumption, as well as signal acquisition and throughput capabilities of the available WSN systems [109]. Figure A.2, shows the building blocks of WiSeMote node as illustrated in [109] with a processing unit, radio unit and an onboard three-axis accelerometer. The processing unit of the WiSeMote is a Texas Instruments MSP430F47187 micro-controller [114]. It is an ultra-low power MCU with a 16-bit RISC based architecture that handles and coordinates all operations occurring in the node at all times, with a maximum clock speed of 16 MHz, 120 KB of flash, 8 KB of RAM, seven Sigma-Delta 16-bit ADC's that can simultaneously sample, and 16-bit data processing [114].

A fast, low power, 256 KB SRAM chip as an external component is used to increase the memory capabilities on the nodes, to store data generated by the node. An external serial flash memory chip is also used. The radio unit is an XBee DigiMesh radio transceiver module with a maximum supported data-rate of 250 kbit/s. The transceiver module is an IEEE 802.15.4 standard compliant [115]. The XBee transceiver has coverage of 100 feet when deployed indoors and with direct line-of-sight in outdoor systems, the

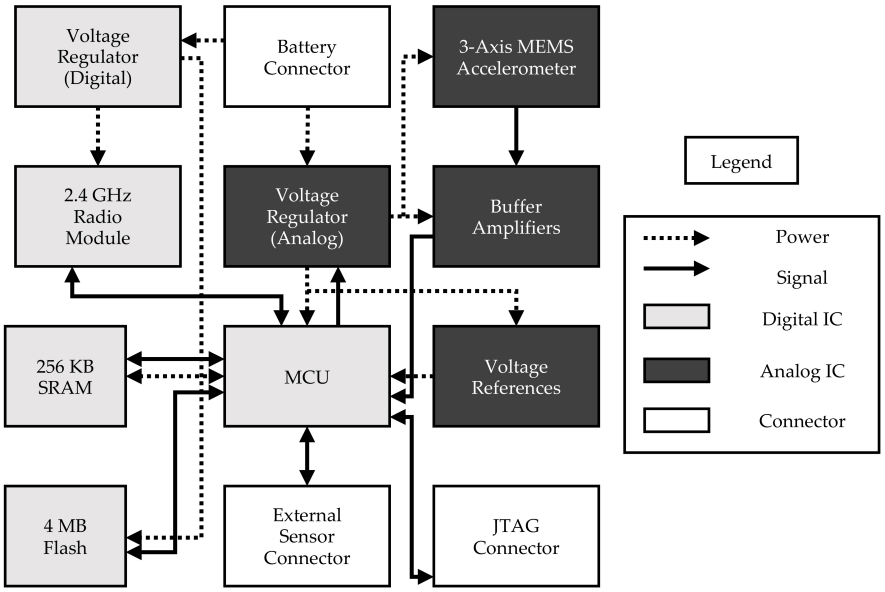

Fig. A.2 Functional block diagram of WiSeMote Node as illustrated in [109]

coverage is up to 300 feet. The transceiver supports ZigBee communication protocol and also a proprietary routing protocol called DigiMesh. Every WiSeMote node has an onboard three-axis accelerometer used for data acquisition [109]. An expansion connector is used for connecting additional sensor suites for data acquisition depending on the application [109].

A 1.76 Ah Lithium-Ion battery is added to the node for its power requirements. It requires a 3.3 V input for operation and the base station is used as a data sink for collecting data from the nodes. The base station has the required components for an external SD card to increase the memory storage [109]. This increases the amount of data that can be stored in the base station [109]. The base station is included with a software for FAT16 driver that provides a better interface with a computer running on a windows operating system. From the published work, the base station was planned to have a GPS module that provides a mechanism to timestamp the data generated from the nodes [109].

A.0.4 Comparison of existing WSN systems

PhyNode is an application-based ultra-low power WSN node. The context of the application is in facility logistics application, which is supposed to be

implemented in warehouses. It is designed with energy neutral operations because of the application requirements provided by the concept of PhyNetLab. The specific requirements from the application, to design the hardware are implemented with additional features. The additional features are used in facilitating a better experimentation experience and to provide data for deep analysis of the hardware operation and energy consumption. A comparison is made in this section between the existing WSN hardware systems to provide hints on the system requirement of PhyNode and to improve the feature set of the hardware with an energy neutral operation.

	WiSeMote	Telos (2005)	Imote2 (2007)	PhyNode
System Current Draw				
Active[1]	$\leq$5 mA	$\leq$15 mA	46 mA	100μA/MHz
Sleep[2]	150 μA	54.5 μA	N/A	-
Hibernate[3]	N/A	5.1 μA	0.5 mA	0.5 μA
micro-controller				
Type	TI MSP430F47187	TI MSP430F1611	X-Scale PXA271	TI MSP430FR5969
Max Clock Speed	16 MHz	8 MHz	416 MHz	16 MHz
RAM Size	8 KB	10 KB	256 KB SRAM 32 MB SDRAM	2 KB SRAM 64 KB FRAM
Flash Size	120 KB	48 KB	32 MB	64 KB
Radio				
Type	XBee DigiMesh	Chipcon CC2420	Chipcon CC2420	TI CC1200 / TI CC2530 SoC
Receive Current	50 mA	19.7 mA	19.7 mA	0.5 mA to 23.5 mA peak at 1.2kbps
Transmit Current	45 mA	17.4 mA	17.4 mA	36 mA (+10dBm) to 46 mA (+14dBm)
Sleep Current	¡50 μA	0.3 μA	0.3 μA	0.12 μA
Wireless Protocol	DigiMesh/ 802.15.4	802.15.4 only	802.15.4 only	802.15.4, ZigBee, RF4CE, 6LoWPAN
External Modules				
RAM Size	256 KB	-	-	N/A
Flash Size	4 MB	1 MB	-	N/A
Battery	Base station	2xAA	3XAAA	Thin Film Battery ¡1mAh / Li-Ion battery 2400mAh
Energy harvesting	N/A	N/A	N/A	BQ25505

1. MCU fully awake, worst case consumption for peripherals, radio asleep.
2. All modules asleep, MCU in idle state.
3. All modules asleep, some modules completely electronically disconnected. Module cannot quickly wake up from sleep.

Table A.1 Comparison of WSN systems [109], [18], [110], [10], [116], [117]

Table A.1 provides an insight into the comparison between existing WSN hardware platforms in terms of the power consumption metrics, wireless communication capabilities using a specific transceiver chip, MCU operation and other external modules. Micro-controllers used in two of three modules are from the MSP430 series, manufactured by Texas Instruments. PhyNode also uses a MSP430 based micro-controller chip. Two of the three WSN nodes in table A.1 use Texas Instruments based radio transceivers. Even though the MCU comes from the same family of ultra-low power electronics, the Ferroelectronic Random Access Memory (FRAM), available in PhyNode, was exclusively selected to provide reduced energy consumption and supersedes the energy consumption ratio over the other micro-controllers [10], [118]. PhyNode uses two types of radio transceivers, one in the 2.4 GHz band and the other in the sub 1 GHz band. They both use Texas Instruments based low-power wireless transceivers CC2350 and CC1200 respectively. External modules used in the WSN systems are supposed to improve the operation and to provide more flexibility in an application.

The hardware selection with the energy harvesting photovoltaic cell clearly shows that PhyNode is capable of energy neutral operations, which is supported with preliminary experiments [10]. It was considered during the design phase that the research platform should not only provide energy neutral capabilities by harnessing the technological advancements in battery technologies and indoor photovoltaic but also provide more than one communication medium for designing application based physical internet entities that are reliable in communication and can coexist with other legacy industrial WSN systems. These were the motivations to build a hardware platform from ground up that facilitated to hold experiments in diverse wireless physical media and also the flexibility to deploy to develop industrial systems and develop network protocols that aid in achieving an energy neutral operation of the sensor node platforms [10].

The third section of the table A.1 provides a detailed comparison of the radio communication capabilities of three WSN systems that operate with the IEEE 802.15.4 standard. This table asserts the fact for testing familiar wireless protocols without the need to access or change the hardware [10]. The last section on table A.1 shows the external modules that are available on the platform. It provides the specifications about the external modules that are interfaced with the processing unit of the WSN platforms described. The important point to note is that WiSeMote is deployed for long term data collection operations and it requires a 4 MB flash memory, whereas PhyNode will be deployed in a materials handling facility, where connections to the routers will be available for instantaneous data transfer and the testbed architecture facilitates access to historical data. The storage for PhyNode is cloud connected and can be interfaced using the RESTful application-programming interface. The data can also be queried from the

router real-time data from the experiment. State-of-the-art distributed cloud computing techniques are used for data storage and data retrieval for the nodes in real-time. Energy harvesting is an external module that helps charging the thin film battery. The power source can be switched between the batteries with the power management integrated circuit (IC).

From the comparisons drawn by listing popular WSN hardware and their system description, a clear trend in the hardware description can be seen that PhyNode provides better hardware components and storage facilities for an indoor WSN implementation. Moreover, energy harvesting capabilities and providing two RF bands for WSN operation make it more desirable for experimentation on various IEEE 802.15.4 networking protocols.

A.0.5 Testbed facilities and research infrastructures

From the sections discussed earlier, there is a clear trend and an imminent revolution in the future of the Internet and it is highly influenced by the emerging Internet of Things and WSNs [119, 120]. There is a factual prediction that the industrial adaptation of Internet of Things is increasing with demand for M2M communication, where industrial processes can be made efficient with these devices and can reduce human intervention in certain processes. PhyNetLab is a concept for increasing the desirability for industrial adaptation of WSN and to usher in new types of WSN hardware with energy neutral design for operation of ultra-low power wireless sensor nodes [11]. A testbed is conceptualized to host energy constrained WSN, because for such implementations, physical analysis is more important than simulation and emulation tools since the dynamics of the environment influence the working of these networks. Dynamics of radio, interference, resource availability and other considered features should be analyzed and modeled before large-scale deployments of WSNs at industries. [11]

Key Internet experimental facilities and WSN infrastructures with existing industrial implementations were surveyed to design the architecture of PhyNetLab. It is a combination of state-of-the-art systems and concepts aimed at implementing systems that are emerging in industrial systems. This is achieved by creating application-based experiments to understand the operation of state-of-the-art information technology in coordination with energy neutral WSNs. Experimental facilities are key instruments that enable practical research which is important with the dynamic characteristics in RF communication [135]. Distributed network measurements are essential to characterise the structure dynamics and operational state of the network, which can be facilitated by an industrial testbed where the application consists of a

Testbed	Scope	Nodes and locality	Noteworthy
MoteLab [121]	WSN	190 TelosB motes, indoor, spread across 3 floors of a laboratory building	One of the first and longest lasting testbeds. In-situ power measurements on some nodes. MoteLab testbed service framework used as a basis for various other testbeds, e.g. CCNY-CWSNET and INDRIYA
NetEye [122]	WSN	130 TelosB motes, indoor, 1 room, 15 wooden benches 1 feet apart	FCFS scheduling approach, static 3db attenuator to each mote antenna for realising multi-hop network and different power levels, to be integrated as part of the Kansai-Geni testbed
TutorNet [123]	WSN	104 motes (91 TelosB, 13 MicaZ), indoor, 1 room	The user indicates the start and end time for its reservation, as well as the list of motes it would like to reserve. The reservation will fail when attempting to exceed the allowed node hours quota, where a node hour represents a reservation unit corresponding to reserving one mote for one hour
MIRAGE / Intel [124]	WSN	100 MicaZ motes (currently only 58 available), indoor, laboratory environment	The reservation is based on an abstract resource specification language for resource discovery (only per-node attribute available) and by using the MIRAGE bidding language in order to reserve the discovered resources
VineLab [125]	WSN	48 TelosB motes, indoor, 1 floor inside a laboratory	unlike other testbeds provides only very basic experimental user support via Python scripts, utilisable for research on smart in-door environments
Kansei [126]	Mesh, WSN	210 XSM motes, 210 Stargate gateways, 50 Trio motes, indoor	One of the most advanced surveyed testbeds with various testbed service functions, co-simulation support, mobility support using mobile robots, event injection possible both at GW and mote level
WISEBED [127]	WSN	750 motes (200 iSense, 143 TelosB, 108 G-Node, 100 MSB-A2, 44 SunSPOT, 60 pacemate, 24 Tnode), inand outdoor, 9 different locations in Europe	Federation architecture, co-simulation support, topology virtualisation, in-situ power measurements on some nodes, mobility support using 40 mobile robots
FRONTS [128]	WSN	21 iSense motes, indoor, laboratory environment	In-situ power measurements on some nodes, purely wireless in-band management
DES-Testbed [129]	Mesh, WSN	95 MSB-A2, 95 Linux nodes, in- and outdoor	Mobility planned, currently one prototype robot available
w-iLab.t Testbed [130]	Mesh, WSN	200 TMoteSky motes, indoor, laboratory environment	Measurement and battery emulation, repeatable mobility support
Senslab [131]	WSN	1024 WSN430 (521 with 802.15.4 MAC, 512 with free MAC layer), indoor, 4 different sites in France	Energy measurement supported for every node, repeatable mobility via electric toy trains
KanseiGeni [132]	Mesh, WSN	576 motes (96 XSM, 384 TelosB, 96 iMote2) attached to Stargate gateways, indoor, laboratory environment	Based on Kansei and Neteye, federation with other GENI envisioned, heterogeneous mote platforms
TWIST [133]	WSN	204 motes (102 TelosB, 102 eyesIFX)	Forms the basis for a variety of other testbeds such as WUSTL

Table A.2 Survey of publicly available WSN testbeds as surveyed by [134] [11]

network with more than two communication stacks [136]. Although in the last decade several such systems have been created, access to these systems remotely has always been a bottleneck as the infrastructure requires complex information technology infrastructure [136]. The ease of access and complex network infrastructure aids in the orchestration of complex measurements [136]. In this section various wireless sensor network testbeds are studied and design guidelines are established for the design of energy neutral WSN testbed. A number of implementations of the testbed with respect to network analysis, WSN hardware analysis, and communication analysis in WSNs were studied, surveyed and reported in [134], after analysis and relevance they are listed in the table A.2. Out of the listed testbeds, three relevant test bed implementations are discussed in detail in A.0.6 as they are considered relevant in multiple silos of PhyNetLab testbed implementations. These features can be used as guidelines for the implementation of an energy neutral testbed with context to applications in industry. This will not only provide results on the methods of energy neutral design adoption in an industrial context but the architecture model can be consolidated into an industrial reference model for the implementation of industrial WSNs. [11]

A.0.6 Relevant work

There is a wide variation of WSN testbeds with specific outcome in the field of WSNs are available as listed in table A.2. Due to a widely accepted research outcome in the field of WSNs, these testbeds have grown into federations. A testbed federation exists, when there is more than one location where the testbed is deployed and coordinated research is carried out in a distributed manner. One of the important aspects of testbeds to grow into federations is that they have matured tools that have been proven to perform with scalability and provide reliable results for the experiments. Gluhak, Alexander, et al's survey of various IoT research facilities provides a deep insight on understanding the various features of the IoT research facility [134], [137] [138]. All of the testbeds have certain features in common, which are provided as foundational elements for designing the PhyNetLab testbed architecture. [11]

A.0.6.1 WISEBED

WISEBED is a WSN research facility, which was first deployed at the University of Lübeck and was later federated across Europe at 8 other

locations. Even though the testbed is not an active project since 2011, the components that the testbed has produced during the runtime of the project has made it a prominent testbed for WSNs research. It was a heterogeneous implementation with each WISEBED partners maintaining its own testbed with different hardware equipment and setup [139]. There was an overlay network that gave access to the testbed as one large, heterogeneous WSN implementation without any geographic disparities for experimentation. Each testbed can be accessed separately to perform experiments in the testbed [127]. It featured an infrastructure of interconnected test beds with more than 8 different platforms to perform highly distributed IoT experiments [127]. The testbed implementation at the University of Lübeck has three types of sensor nodes with IEEE 802.15.4 radio interface for experimentation. The testbed provided API endpoints for data acquisition for the user and provided a testbed runtime which included the clients for the API access that could be used for acquiring the data from the testbed experiments. [11]

The software components of the testbed include a testbed runtime, client libraries for consuming endpoints from the testbed, a library for embedded devices in the testbed. The following are listed in the web documentation of WISELIB and articles that describe the same [140], [139]. An operating system called LorienOS with unparalleled modularity to adapt WSN applications. The testbed has a Java based runtime to perform the functions of the testbed. It is an Apache Maven repository that can be installed when new testbed installations are deployed. One of the many notable outcomes of the testbed was a generic library called WISELIB [139]. The WISELIB is a library of algorithms for heterogeneous WSN embedded devices. It contains various algorithm classes such as localization and routing that can be compiled for several platforms such as iSense or ContikiOS platforms, or for WSN simulators (example: shawn). WISELIB is written in C++, with the same approach as Boost and CGAL, which uses templates to improve adapting the features provided by the library [139]. This makes it possible to write generic and platform independent code that is very efficiently compiled for the various platforms [139]. [11]

A.0.6.2 FIT - IoT

FIT - IoT is a testbed, which also started out as a WSNs testbed and was deployed in more locations, which later became the federated testbed for wireless sensor networks called Future Internet of Things (FIT - IoT Lab). It is a large-scale research facility with state-of-the-art technologies deployed over 6 locations with more than 2000 nodes collectively. There are autonomous vehicles that have a planned trajectory to simulate scenarios of

mobility in wireless sensor nodes. The hardware platform of FIT - IoT testbed was called WSN430, which is powered with the MSP430 family of micro-controllers from Texas Instruments. To provide multi-user context for experimenting on the platform there is an approach using user virtual machines. It provides a command line interfaces for each user to interact with nodes and to perform different actions depending on the experiment being performed [131]. Unfortunately, with an increasing number of users, scalability of infrastructure becomes expensive with virtual machines in terms of hardware and resources used. FIT - IoT currently hosts three types of node classes namely open node, host nodes and the control node. The open node gives a full IoT experimentation environment, which can be compared to the PhyNode in PhyNetLab [11]. These nodes are the sensor nodes in the testbed, in a testbed context these nodes are the nodes that are deployed for experimentation. The host node and the control node can interact with the open node actively or passively [141]. [11]

A.0.6.3 Indriya

Indriya is a low-cost, 3D WSN testbed which has teslob nodes deployed in a testbed. It was implemented at the National University of Singapore with 139 wireless nodes that are connected with active-USB cables [19]. The approach to provide dynamic reprogramming of the nodes in the testbed was considered in this testbed, which was one of the reasons to implement a secondary radio device that provides flexible interaction with the sensor nodes. The hardware platform with its interconnections and node intra-connections to facilitate experimentation in the testbed is described in the chapter PhyNetLab hardware systems. In the testbed Indriya, an active-USB cable infrastructure provides a back channel for remote programming and also for powering the sensor nodes [19]. This testbed in many ways is a low cost implementation of Motelab from Harvard university. Its low cost implementation with limited requirement for maintenance due to the power source and flexible reprogramming using the active- USB cables is considered an important factor of relevance to be considered for PhyNetLab design. Even though it provides flexibility via the cables, it reduces the mobility of the nodes and increases the requirements on the testbed infrastructure. [11]

WISEBED, FIT/IoT-LAB and Indriya are the three relevant testbed architectures that are considered for design of PhyNetLab testbed architecture. Since these testbeds have grown into a federation of testbeds or have proven to perform better in context to WSNs. [11]