NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

JOÃO RICARDO BOGALHO BRILHA

BSc in Computer Science and Engineering

MICROBABEL: A MULTI-RADIO ARCHITECTURE FOR RESILIENT AND DECENTRALIZED IOT NETWORKS

MASTER IN COMPUTER SCIENCE AND ENGINEERING

SPECIALIZATION IN DISTRIBUTED AND PARALLEL SYSTEMS

NOVA University Lisbon

Draft: *February 6, 2026*

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

MICROBABEL: A MULTI-RADIO ARCHITECTURE FOR RESILIENT AND DECENTRALIZED IOT NETWORKS

JOÃO RICARDO BOGALHO BRILHA

BSc in Computer Science and Engineering

Adviser: João Leitão
Associate Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE AND ENGINEERING
SPECIALIZATION IN DISTRIBUTED AND PARALLEL SYSTEMS

NOVA University Lisbon
Draft: *February 6, 2026*

Abstract

Nowadays, small sensors and actuators can form the basis for sophisticated technological solutions for Internet of Things (IoT) infrastructures and domotics applications that monitor and control spaces, or even promote safety in case of disasters, by providing guidance in a reactive and timely manner. Unfortunately, existing commercial solutions in the domains of IoT and domotics are dependent on centralized and cloud-based infrastructures to operate, which at best can lead to the unavailability of these solutions when connectivity is lost, and at worst – particularly in the context of safety-critical space management – potential human loss, as infrastructures can easily become inaccessible during a disaster.

One way to address this is to take advantage of direct interactions between devices and small on-premises edge servers, exploiting the myriad wireless technologies (i.e., Wi-Fi, Bluetooth Low Energy (BLE), LoRa, ZigBee, ESP-NOW) that allow such systems to remain available even when centralized infrastructures or Internet access are not. One can leverage multiple of these technologies to make IoT and domotics systems more robust, by automatically selecting protocols based on their availability per device, medium congestion, among other factors. However, taking advantage of this heterogeneity is challenging: on one hand, because these devices usually have limited computational power, and on the other, because tackling this level of complexity is a hard task for application developers.

To overcome these challenges, in this work we plan to study and develop mechanisms to automatically exploit that heterogeneity – essentially a specialized variant of the Babel framework (originally developed to assist in the development of distributed protocols and applications) for small and integrated devices, tentatively named **μ-Babel**. We plan to validate and evaluate our solution through two relevant use cases: one focused on data acquisition within a large area (e.g., university campus), and another focused on a disaster-response system in the context of a building, combining data acquisition, local decisions, and actuation through alerts and emergency guidance.

Keywords: Internet of Things (IoT) · Autonomous Operation · Multi-Radio Communication · Disaster Resilience · Embedded Systems · Peer-to-peer (P2P) Mesh Networks

Resumo

Atualmente, pequenos sensores e atuadores podem formar a base de soluções tecnológicas sofisticadas para infraestruturas Internet of Things (IoT) e aplicações de domótica que monitorizam e controlam espaços, ou até promovem segurança em caso de desastres, fornecendo orientação reativa e atempada. Infelizmente, as soluções comerciais existentes nos domínios de IoT e domótica dependem de infraestruturas centralizadas e baseadas na *cloud* para operar, o que na melhor das hipóteses pode levar à indisponibilidade destas soluções quando a conectividade é perdida, e na pior – em particular no contexto de gestão de espaços críticos para a segurança – potencial perda humana, uma vez que as infraestruturas podem facilmente tornar-se inacessíveis durante um desastre.

Uma maneira de abordar esta questão é tirar partido de interações diretas entre dispositivos e pequenos servidores *edge* locais, explorando as diversas tecnologias sem fios (i.e., Wi-Fi, Bluetooth Low Energy (BLE), LoRa, ZigBee, ESP-NOW) que permitem que tais sistemas permaneçam disponíveis mesmo quando infraestruturas centralizadas ou acesso à Internet não estão. É possível utilizar várias destas tecnologias para tornar os sistemas IoT e de domótica mais robustos, selecionando automaticamente protocolos com base na sua disponibilidade por dispositivo, congestão do meio, entre outros fatores. No entanto, tirar partido desta heterogeneidade é desafiante: por um lado, porque estes dispositivos normalmente têm poder computacional limitado, e por outro, porque lidar com este nível de complexidade é uma tarefa difícil para os programadores de aplicações.

Para superar estes desafios, neste trabalho planeamos estudar e desenvolver mecanismos para explorar automaticamente essa heterogeneidade – essencialmente uma variante especializada da framework Babel (originalmente desenvolvida para auxiliar no desenvolvimento de protocolos e aplicações distribuídos) para dispositivos pequenos e integrados, tentativamente denominada μ**-Babel**. Planeamos validar e avaliar a nossa solução através de dois casos de uso relevantes: um focado na aquisição de dados numa área ampla (e.g., campus universitário), e outro focado num sistema de resposta a desastres no contexto de um edifício, combinando aquisição de dados, decisões locais, e atuação através de alertas e orientação de emergência.

Palavras-chave: Internet of Things (IoT) · Operação Autónoma · Comunicação Multi-Rádio · Resiliência a Desastres · Sistemas Embutidos · Redes Mesh Peer-to-peer (P2P)

Contents

List of Figures

List of Tables

Glossary

ESP-NOW Wireless communication protocol developed by Espressif. Enables direct and low-power communication between ESP-family devices without the need for a router. *(pp. i, ii, 4, 8, 14, 26, 30, 34, 35)*

LoRa From "long range", proprietary radio communication technology *(pp. i, ii, 3, 4, 8, 13, 18, 20, 25, 26, 29, 30, 34)*

Modbus An application-layer client/server serial communication protocol, widely used in industrial automation systems. *(p. 14)*

MQTT Lightweight publish/subscribe messaging protocol designed for constrained devices and low-bandwidth networks. Previously stood for "Message Queuing Telemetry Transport", but since 2013 it does not stand for anything in particular *(pp. 7, 15, 23, 27, 34)*

Thread Low-power, mesh networking technology based on the IEEE 802.15.4 standard. Increasingly adopted for smart-home applications. *(p. 26)*

μ-Babel MicroBabel *(pp. i, ii, v, 2–4, 6–8, 19, 20, 22–28, 30, 32, 33)*

ZigBee Low-power, low-data rate wireless communication specification based on the IEEE 802.15.4 standard. Designed for short-range communication in mesh, star or tree network topologies. *(pp. i, ii, 4, 8, 14, 15, 26, 30)*

Acronyms

AES Advanced Encryption Standard *(pp. 14, 16, 18, 19)*
API Application programming interface *(p. 28)*
AWCT Always Connected Things *(p. 13)*

BLE Bluetooth Low Energy *(pp. i, ii, 3, 4, 8, 12–15, 18, 20, 25, 26, 28–30, 34, 35)*

CA Certificate authority *(p. 6)*
CoAP Constrained Application Protocol *(pp. 7, 23, 27)*
CPU Central processing unit *(pp. 25, 27)*

DNS Domain Name System *(p. 1)*
DODAG Destination-oriented directed acyclic graph *(p. 11)*
DTN Delay-tolerant networking *(pp. 11, 14)*

ESP-IDF Espressif IoT Development Framework *(p. 27)*

FHSS Frequency-hopping spread spectrum *(pp. 14, 16)*
FR Functional Requirement *(pp. 22, 34)*

GASE Group Authentication Scheme at the Edge *(p. 19)*
GPIO General-purpose input/output *(pp. 28, 32)*
GPS Global Positioning System *(p. 18)*

HAL Hardware Abstraction Layer *(pp. 28, 32)*
HF High frequency *(p. 14)*
HTTP Hypertext Transfer Protocol *(p. 11)*
HVAC Heating, ventilation, and air conditioning *(p. 5)*

I^2C Inter-Integrated Circuit *(pp. 28, 32)*

IoT Internet of Things *(pp. i–iii, 1–3, 5–13, 18, 19, 21, 23, 24, 26)*

KDC Key distribution center *(p. 6)*

LE Low Energy *(p. 26)*
LoRaWAN LoRa Wide-Area Network *(pp. 9, 13, 15)*
LPKM Lightweight Polynomial-based Key Management *(p. 19)*
LPWAN Low-Power Wide-Area Network *(p. 13)*

MAC Message Authentication Code *(p. 19)*
ML Machine learning *(p. 15)*

NFR Non-Functional Requirement *(pp. 7, 22, 23, 34)*
NTP Network Time Protocol *(p. 18)*
NVIS Near Vertical Incidence Skywave *(pp. 14, 15)*

OS Operating system *(p. 27)*
OSPF Open Shortest Path First *(p. 23)*

P2P Peer-to-peer *(pp. i, iii, iv, 3, 4, 6, 9–12, 22, 24–26, 34)*
PKI Public key infrastructure *(p. 19)*
PLR Packet loss ratio *(p. 10)*
PRNG Pseudorandom number generator *(pp. 14, 16)*
PSRAM Pseudostatic random-access memory *(p. 25)*
PUF Physically Unclonable Function *(pp. 19, 22)*

QoS Quality of Service *(pp. 4, 6)*

RAM Random-access memory *(p. 27)*
RF Radio frequency *(p. 13)*
RGCS Randomized gossip-consensus-based sync *(pp. 17, 18, 20, 30)*
RPL Routing Protocol for Low-Power and Lossy Networks *(pp. 10–12, 14)*
RSSI Received signal strength indicator *(p. 14)*
RTOS Real-time operating system *(pp. 27, 28)*

SBC Single-board computer *(p. 26)*
SDK Software Development Kit *(p. 27)*
SDN Software-defined networking *(pp. 6, 15, 20, 21)*
SF Spreading Factor *(p. 8)*
SPI Serial Peripheral Interface *(pp. 26, 28, 32)*

SRAM Static random-access memory *(p. 25)*
SSID Service Set Identifier *(p. 8)*

TCP Transmission Control Protocol *(pp. 6, 10, 12, 20)*

UART Universal asynchronous receiver-transmitter *(p. 28)*
UDP User Datagram Protocol *(p. 7)*
UUID Universally unique identifier *(pp. 29, 30)*

WSN Wireless Sensor Network *(pp. 10, 14, 16, 17)*

1

Introduction

The proliferation of Internet of Things (IoT) devices has changed how we monitor and interact with physical spaces – from smart homes to industrial facilities – with the global IoT market reaching 18.5 billion connected devices in 2024, and projected to reach 39 billion by 2030 [40], with industry analysts estimating that IoT technologies could unlock between $5.5 and $12.6 trillion in economic value by the same year [9].

However, current IoT systems remain fundamentally dependent on continuous Internet connectivity and centralized cloud infrastructures [50, 19]. The dominant architectural pattern across the industry relies on edge devices collecting data and transmitting it to remote cloud servers for processing, storage, and execution of other control logic. This approach delivers scalability and ease of management but creates a fundamental dependency on network availability.

This centralized model introduces several concerns beyond just connectivity: operational costs for cloud services create economic dependencies on third-party providers and often lead to vendor lock-in, data sovereignty issues arise when sensitive information must travel to and reside on external infrastructure, and system resilience becomes fundamentally tied to the availability of these remote services.

While cloud platforms can benefit IoT deployments by providing additional computational resources and storage capacity, systems that *depend* on constant cloud connectivity sacrifice local autonomy and introduce single points of failure. This cloud-centric model has enabled rapid IoT adoption, but introduces critical vulnerabilities across scenarios where continuous connectivity cannot be guaranteed.

Recent infrastructure failures illustrate the fragility of cloud-dependent architectures: the AWS US-EAST-1 outage in October 2025 [42] disrupted services from banking to smart homes worldwide, demonstrating how a Domain Name System (DNS) configuration error in a *single region* can have cascading effects, leading to global disruptions [41].

Attempts have been made to address these challenges through incremental improvements, such as shifting focus to edge computing in order to reduce latency [19, 50, 36], redundant cloud regions for availability [28, 27], and hybrid architectures that combine local and remote processing [1, 11, 22].

However, these solutions remain fundamentally tied to the assumption of eventual connectivity, and often increase system complexity without eliminating the core dependency. This leads to shortcomings in key areas that demand more fundamental architectural reconsideration:

Limited suitability for hazardous environments: Remote or dangerous locations (industrial sites, disaster-prone areas) require systems that can operate reliably without constant human intervention or stable network infrastructure;

Lack of autonomous operation: Device deployment and operation en masse can be brittle, with little tolerance for individual node failures in the IoT infrastructure, which is naturally susceptible to network failures and intermittent connectivity;

Privacy and data sovereignty concerns: Cloud platforms and other third-parties are oftentimes an unavoidable middle layer between end devices and end users, raising questions about data processing and control.

While these challenges affect both IoT and domotics systems alike – from industrial monitoring to residential automation – they become particularly acute in disaster response and emergency scenarios, where communication infrastructure might become completely unavailable precisely when needed most, rapid deployment with minimal configuration becomes essential, and autonomous operation transitions from desirable to indispensable.

During earthquakes, floods, and natural or human-driven disasters, the need for real-time sensor data (structural integrity, air quality, evacuation routes) and bidirectional communication (threat alerts, user feedback) becomes critical, yet traditional infrastructure often fails first, eliminating both cloud connectivity and local network access points that deployed devices rely on, making support systems for these scenarios unavailable or non-operational.

A pragmatic issue in the context of smart homes is that without cloud connectivity, users cannot interact with their appliances, such that during a Wi-Fi outage smart lights become uncontrollable despite all hardware being physically present.

While merely inconvenient domestically, this architectural dependency has critical implications in other contexts and **μ-Babel** aims at addressing such scenarios across different application domains: from residential systems requiring local control, to building monitoring infrastructures that must operate during emergencies, to disaster response networks where autonomous operation becomes essential when traditional infrastructure fails.

While cloud platforms like Amazon Alexa, Google Home, and Microsoft Azure IoT Hub offer convenience for data processing and remote device control, their inherent dependence on continuous Internet connectivity introduces some critical limitations: increased latency from round-trip communication to distant servers [36, 33], reduced availability during network disruptions or cloud outages [47, 27], and poor fault-tolerance when infrastructure fails [3].

These characteristics make cloud-centric architectures fundamentally unsuitable for scenarios that require or prioritize local operation, autonomous behavior, or guaranteed responsiveness during emergencies.

REMOVED para o chap1 não passar da pág. 4

The Babel Ecosystem [16] partially addresses these limitations by enabling devices to operate autonomously without cloud infrastructure, while still supporting cloud integration when connectivity is available and desired.

In this document we propose **μ-Babel**, a lightweight framework targeting embedded platforms (e.g., ESP32, Raspberry Pi Pico) aimed at developing resilient, multi-radio, and decentralized IoT systems that can operate autonomously during infrastructure failures.

μ-Babel will integrate with the broader Babel Ecosystem, which runs on hardware with greater resources such as full Raspberry Pi boards, desktops/laptops, or servers. This enables a heterogeneous architecture where resource-constrained edge devices can seamlessly interoperate with computational nodes for additional data aggregation, processing, and large-scale coordination.

FIXED: "class-based"

FIXED: multi-protocol -> multi-radio mas dar atenção à literatura

Note on terminology: Throughout this work, the term *multi-radio* refers to heterogeneity in communication technologies (i.e., Bluetooth Low Energy (BLE), LoRa, Wi-Fi, etc.) operating in distinct link/physical layers. In the literature, *multi-protocol* is commonly used to describe the same concept, but we try to avoid this nomenclature where possible given our usage of *protocol* as a distributed application (see Section 3.6). Neither of these terms should be confused with *multi-channel*, which refers to frequency diversity within a single radio technology stack (e.g., 2.4GHz Wi-Fi channels 1-13, LoRa frequency hopping).

TODO: validar esta definição please

Main Research Questions

In this work, we focus on three main research questions:

How can these systems maintain communication and full operation when traditional infrastructure fails?

Conventional deployments of networked devices often heavily rely on Wi-Fi access points, cellular towers, or other centralized infrastructure that can easily become unavailable during disasters, or is altogether unreliable in remote locations.

μ-Babel will address this by leveraging a multi-radio communication approach, supporting a diverse set of wireless protocols that can operate independently of infrastructure.

By enabling adaptive radio interface selection and Peer-to-peer (P2P) mesh formation, devices can establish alternative communication paths when primary channels fail, allowing them to sustain information exchange and system availability.

FIXED: muito ênfase em iot

How can device heterogeneity be leveraged to create and orchestrate these systems?

Deployments naturally comprise devices with varying capabilities, from simple resource-constrained nodes, to more capable gateways that might act as aggregators or bridges. Rather than treating this heterogeneity as a limitation, **μ-Babel** will exploit it through capability-aware protocols that allow devices to negotiate roles dynamically via autonomous and automatic discovery.

Resource-rich nodes can serve as data aggregation and processing points, or bridges between a remote deployment and traditional network infrastructure (if desired), while simpler devices focus on sensing and actuation, creating a resilient multi-tier architecture.

How can we achieve (near-)zero-configuration deployment in diverse environments? Straightforward deployments are essential where users cannot perform complex configurations – from smart homes and factories to hazardous areas. **μ-Babel** will provide automatic peer discovery across multiple radio interfaces, self-organizing network formation, and decentralized coordination mechanisms that eliminate the need for pre-configured coordinator nodes or manual network planning. Devices will autonomously establish connectivity with each other, negotiate protocols, and begin operation upon being activated, with minimal configuration effort to enable rapid deployment even in hard-to-reach locations.

FIXED: demasiado foco em emergências

Expected Contributions

We plan to make the following main contributions:

- A decentralized architecture supporting multiple communication technologies (Wi-Fi, BLE, LoRa, ZigBee, ESP-NOW) with adaptive switching based on Quality of Service (QoS) requirements, resource availability and device capabilities;
- A resource-efficient programming framework for embedded platforms that enables autonomous operation without central coordination, providing abstractions for multi-radio communication, peer discovery, and opportunistic data forwarding;
- A proof-of-concept implementation demonstrating infrastructure- independent operation and automatic disaster-mode failover in a real-world deployment.

Document Structure (Roadmap)

FIXED: added doc structure

The remainder of this document is arranged as follows:

Chapter 2 Reviews widely-used communication protocols and related work in P2P networking, multi-radio communication, decentralized synchronization, and establishes the scope of this work in regard to security.

Chapter 3 Presents the system requirements, operating conditions, failure modes, hardware and software platforms, and the architectural model of the proposed solution.

Chapter 4 Summarizes the current implementation status and outlines the development and evaluation plan.

2

Related Work

Internet of Things (IoT) and home automation (domotics) systems share the same technological building blocks (wireless sensors, embedded devices, network communication) but diverge significantly in their operational focus and architectures:

FIXED: added citations

IoT systems [12] typically focus on **data collection and monitoring**, by streaming sensor readings to centralized platforms for analysis and processing, with control functions often being a secondary concern.

Domotics systems [17, 14] instead prioritize **real-time control and actuation** over physical spaces, where responsiveness and local autonomy are paramount for end user experience and privacy.

This distinction gains additional relevance when focusing on resilience requirements: while IoT deployments may tolerate delayed data aggregation and/or temporary connectivity issues in monitoring scenarios, domotics applications (such as emergency lighting control or Heating, ventilation, and air conditioning (HVAC) management) demand immediate local response regardless of network conditions.

Both domains, however, suffer from a common vulnerability when confronted with infrastructure failures during disasters: their predominantly cloud-centric architectures collapse precisely when autonomous operation becomes indispensable.

In the remainder of this work, these two domains are collectively referred to as Cyber-physical systems (CPSs), to emphasize their shared integration of networked computation, sensing, and physical actuation.

FIXED? umbrella term, ainda não atualizei o resto do doc, pending approval

The challenges faced by disaster-resilient CPSs span multiple dimensions:

Infrastructure failures eliminate access points that devices depend on for coordination and operation;

Intermittent connectivity creates network partitions where subgroups of devices must operate autonomously without global state synchronization;

Resource constraints limit the computational, memory and energy budgets available for implementing sophisticated resilience mechanisms in embedded platforms.

These challenges are fundamentally architectural: existing IoT and domotics systems are designed around the assumption of stable infrastructure, treating network partitions and coordinator failures as transient anomalies rather than the norm.

Cloud-centric architectures place control decisions in remote servers, creating dependencies that cannot be satisfied when connectivity fails. Coordinator-based topologies – whether using dedicated gateways, master nodes, Software-defined networking (SDN) controllers, among others – concentrate the risk of failure in single points that can paralyze entire systems if compromised or disconnected. Even ostensibly distributed systems often rely on persistent connections (i.e., Transmission Control Protocol (TCP)) over a single medium, or heavy middleware platforms that exceed the capabilities of embedded devices and, once more, assume some form of infrastructure availability.

The fundamental point of contention is between resilience requirements (autonomy during infrastructure collapse) and resource constraints (limited computation, memory, and energy). Traditional approaches address one at the cost of the other: cloud platforms provide sophisticated coordination but fail during outages; purely local systems avoid external dependencies but struggle with inter-device coordination and protocol heterogeneity with low resource usage.

TOFIX: general statement.. não encontro citations to back this up exactly, sinto que é kind of a given

Disaster-resilient systems require architectural choices that prioritize autonomous Peer-to-peer (P2P) connectivity as the baseline mode of operation rather than having it as a secondary – even exceptional – fallback mechanism.

Naturally, this requires a fundamental rethinking of IoT systems design: how devices discover and communicate with each other using diverse protocol stacks (without depending on centralized control), how they achieve temporal synchronization and coordination without master beacons, how to optimize the usage of limited available resources, and how to maintain secure operation throughout without persistent access to inherently centralized components such as Certificate authorities (CAs) or Key distribution centers (KDCs).

The following sections briefly cover related work across four areas that collectively enable autonomous operation: (Section 2.1) common wireless communication technologies across application, link, and physical layers; (Section 2.2) P2P mesh networking and topology management for autonomous network formation without coordinator dependencies; (Section 2.3) multi-protocol communication and adaptive selection based on Quality of Service (QoS) and device capabilities; (Section 2.4) decentralized synchronization mechanisms for coordinating multi-protocol communication without coordinator nodes.

For each area, we examine how existing approaches handle (or fail to handle) infrastructure failure scenarios, and identify architectural assumptions that conflict with

disaster-resilient requirements.

Additionally, security and privacy considerations are presented in Section 2.5, which outlines the scope of this work in regard to higher-layer safety mechanisms.

FIXED: separei a section sobre security, no longer part of "autonomous operation" areas per se

2.1 Wireless Communication Technologies

A core pillar of resilient and decentralized IoT systems is the heterogeneity of available communication capabilities and wireless protocols across different layers of the communication stack. Different technologies naturally expose different trade-offs in terms of infrastructure requirements, range, throughput, discovery mechanisms, and energy consumption – all of which directly influence the design choices around their usage.

FIXED: moved protocols para aqui + MQTT and CoAP overview

There is no single communication protocol that fully addresses the requirements of infrastructure-free and disaster-resilient operation. As such, it is important to establish the capabilities and limitations of each one, as well as their potential for harmonious cooperation within a unified system.

2.1.1 Application-Layer Protocols

Application-layer IoT protocols typically focus on interoperability and efficient data exchange under stable connectivity assumptions rather than autonomous operation during infrastructure loss. At the application layer, two messaging protocols stand out given their wide-spread adoption in IoT deployments:

MQTT: A lightweight publish/subscribe messaging protocol designed for constrained devices and unreliable, low-bandwidth networks. It relies on a centralized *broker* to mediate communication between publishers and subscribers – essential, a server.

Constrained Application Protocol (CoAP) [7, 39]: A RESTful, request/response protocol designed for constrained environments. Operates over User Datagram Protocol (UDP) and assumes the presence of well-known endpoints or proxy servers for resource discovery and access.

Both protocols inherently assume the availability of stable endpoints, and rely on continuous network connectivity in order to operate. These assumptions conflict with the infrastructure-less and failure-prone environments targeted by **μ-Babel**, and emphasize the contrast between higher-layer protocols which tend to optimize for usage under stable infrastructure, and resilience requirements that push architectural decisions towards lower layers.

added narrative connector

For this reason, while such protocols are of interest for broader interoperability (see Non-Functional Requirements (NFRs) in Section 3.2.1), they are not a part of the foundational protocols employed by our work.

2.1.2 Physical- and Link-Layer Technologies

This section reviews the physical- and link-layer wireless technologies relevant to resilient and decentralized IoT deployments, as they are commonly adopted in due to their low energy consumption and hardware availability. While the previous section addressed higher-layer messaging protocols, the technologies discussed here provide the foundational communication stack that determines the behavior of **μ-Babel**.

Table 2.1 details the basic operational characteristics of the wireless communication technologies considered in this work and how they drive protocol selection based on different factors:

Table 2.1: Protocol Characteristics

Protocol	Discovery	Range[1]	Frequency	Data Rate	Power Draw
Wi-Fi	SSID scan	50-100 m	2.4 GHz	1-50 Mbps	High
BLE	Advertising	10-50 m	2.4 GHz	1-2 Mbps	Very Low
ESP-NOW	Peer list	50-100 m	2.4 GHz	1 Mbps	Low
ZigBee	Beacon	10-100 m	2.4 GHz	250 Kbps	Very Low
LoRa	Preamble detection	2-10 km	433/868/915 MHz[2]	0.3-50 Kbps[3]	Low[3]

[1] Outdoor line-of-sight; indoor ranges typically 30-50% lower
[2] Frequency depends on regional regulations
[3] Data rate and power consumption depend on Spreading Factor (SF); higher SF = longer range, lower rate, higher energy cost

get actual milliamp power draw values from somewhere and/or source?

Range vs. Throughput Trade-offs: LoRa provides 2-10 km range (with appropriate antenna configurations) but very limited throughput (0.3-50 Kbps), suitable for sparse periodic synchronization between distant partitions. Bluetooth Low Energy (BLE) and ESP-NOW offer moderate range (50-100 m) with higher throughput (1-2 Mbps), appropriate for dense local deployments where multi-hop forwarding is feasible and often required.

Discovery Mechanisms: Different protocols employ distinct peer discovery approaches. BLE advertising enables continuous passive discovery, ESP-NOW requires explicit peer lists, and ZigBee uses beacon-based association. **μ-Babel**'s hybrid discovery mechanism (Section 3.6.2) will accommodate these differences while translating protocol-specific information into standardized peer descriptors.

Energy Considerations: Protocol selection must account for energy costs: Wi-Fi consumes significantly more power than BLE, while LoRa power consumption depends on SF configuration.

Together, these protocol characteristics illustrate the inherent trade-offs faced by resilient IoT deployments operating without reliable infrastructure. Understanding how

range, throughput, discovery mechanisms and energy consumption vary across communication technologies is essential to establish an architecture that can adaptively leverage multiple protocols instead of relying on a single one.

2.1.3 Discussion

The considered technologies highlight that no single protocol can simultaneously optimize for range, throughput and energy efficiency upon loss of infrastructure. Instead, resilience emerges from the ability to combine the complementary features of each technology and adapt communication strategies to the specific context and requirements of a given environment.

These observations motivate an approach in which diversity is treated as a first-class requirement, and provide the technological context for the multi-radio operation and architectural choices discussed in Chapter 3.

2.2 P2P Mesh Networking and Topology Management

While communication technologies define what devices can transmit, network topologies determine how they organize themselves once infrastructure disappears.

added narrative connector

Infrastructure-dependent star topologies in which devices communicate via a central coordinator or gateway suffer from a similar downside when compared with single-radio communication: when that coordinator becomes unavailable or unreachable, the entire network loses connectivity.

This pattern pervades current IoT deployments, from Wi-Fi access point dependencies to LoRa Wide-Area Network (LoRaWAN) gateway requirements, and becomes catastrophic in disaster scenarios where central coordinators are most likely to fail first, due to power outages, physical damage, or severe network congestion from increased emergency communications traffic.

FIXED: justify this claim

P2P mesh architectures address this limitation by distributing coordination across all participating nodes, thus eliminating single points of failure. Nonetheless, achieving robust mesh operation requires solving three interconnected challenges: neighbor discovery and establishment of initial connectivity, topology maintenance as nodes join/leave a network, and efficient data routing through multi-hop paths when direct communication becomes impossible or inefficient.

2.2.1 Peer Discovery and Topology Maintenance

Tree-based topologies are a common design choice in resource-constrained IoT networks due to their routing simplicity and low coordination overhead. In the Multi-Protocol IoT Gateway implementation presented in [21], devices organize hierarchically with a central gateway serving as the root of three.

added narrative device (cake analogy) aka more focus on the technical aspect of topology management

While that work focuses on multi-radio integration (discussed further in Section 2.3.2), its topology management reveals a limitation of its tree-based approach: the system implements automatic parent reselection when coordinators fail, but the underlying tree structure imposes that nodes can only communicate through their parent-child relationships rather than arbitrary peer connections.

Testing demonstrated successful multi-hop operation up to 5 hops in office environments, but the dependence on linear topologies and the restriction of parent-child communication limit route diversity and resilience, creating dependency chains where a single intermediate node failure can disconnect entire subtrees, rather than behaving as a P2P mesh.

another connector cake analogy thing

Hybrid gossip–flooding approaches can be used to balance overlay adaptability with deterministic broadcast guarantees in P2P networks, and a more sophisticated approach to membership management is presented in HyParView [23]. In this protocol each node maintains two distinct partial views for scalability: a small *active view* (size = fanout + 1) containing nodes with which symmetric links are actively maintained, and a larger *passive view* serving as a backup pool of potential neighbors that may be promoted to the *active view* if one of its nodes fails.

The *active view* is managed reactively, such that nodes are added during join operations and removed upon failure detection, while the *passive view* is maintained cyclically through periodic shuffle operations that randomly exchange node identifiers between peers.

This hybrid strategy enables remarkable resilience, with the system recovering from 80% node failures with minimal reliability loss (maintaining 95% reliability) and from 50% failures in just 1-2 membership cycles, compared to 60+ cycles required by purely cyclic protocols like Cyclon [45].

HyParView's deterministic flooding approach, by broadcasting messages along the entire active view graph rather than probabilistic neighbor selection, enables fast failure detection since every active link is tested at each broadcast. The symmetric link requirement ensures bidirectionality: if node A can reach node B, then B can reach A, preventing the formation of one-way communication paths that complicate routing.

However, HyParView assumes TCP availability for maintaining persistent connections and using connection failures as implicit failure detectors. Additionally, it assumes that any node can communicate with any other node in its active view, which in practice requires some form of routing infrastructure.

This dependency on full network stack functionality makes direct application to resource-constrained embedded platforms challenging, though the architectural principles of hybrid views and shuffle-based passive view maintenance remain valuable.

The heterogeneous disaster IoT architecture [34] discussed in Section 2.3.1 employs Routing Protocol for Low-Power and Lossy Networks (RPL) for its Wireless Sensor Network (WSN) layer, demonstrating practical routing in resource-constrained disaster scenarios.

Their performance analysis reveals considerable trade-offs: convergence time scales linearly from 7 seconds for 20 nodes to 14.5 seconds for 100 nodes, while Packet loss ratio (PLR) at 10 hops reaches 80% with a 10-second sending interval but becomes acceptable at 20-second intervals.

These measurements highlight the throughput limitations imposed by tree topologies, in that their Instance 1 traffic (human data) must be restricted to 1-hop from the root node to ensure the least possible delay in its delivery, defeating the purpose of multi-hop mesh for critical communications.

The Delay-tolerant networking (DTN) component using mobile drones as data mules provides an alternative path for partitioned networks, with a maximum of 20 nodes per Destination-oriented directed acyclic graph (DODAG) to maintain acceptable convergence times during emergency situations, but this approach trades latency for eventual delivery rather than real-time mesh routing.

These approaches reveal fundamental trade-offs: tree-based protocols like RPL offer structured routing for resource-constrained devices but create single points of failure; fully distributed protocols like HyParView achieve resilience through P2P overlay management but assume network capabilities impractical for embedded platforms; and hybrid multi-layer approaches demonstrate practical deployment but still face throughput limitations.

This gap between resilience requirements and embedded capabilities motivates the exploration of topology management strategies more suitable for autonomous operation, particularly in unstable network conditions.

FIXED: add take-away for the section + set up narrativo "motivates exploration..."

2.2.2 Routing in Partitioned and Intermittently Connected Networks

When network partitions prevent end-to-end paths, store-and-forward mechanisms enable eventual data delivery. The Bundle Protocol [37] addresses delay-tolerant networking through custody-based retransmission and opportunistic connectivity exploitation. While the protocol specification predates modern IoT deployments and was not designed specifically for resource-constrained devices, its core principles inform contemporary DTN approaches.

The framework presented in [29] implements elastic bandwidth utilization by dynamically adjusting transmission rates based on available connectivity, and supports scheduled, predicted, and opportunistic transmission windows, taking inspiration from the Bundle Protocol.

Data parcels are compressed, encrypted, and bundled before transmission, with a load balancer managing concurrent transfer threads to optimize bandwidth usage during brief connectivity windows.

While their Hypertext Transfer Protocol (HTTP)-based implementation targets cloud-backed IoT deployments, the core concepts of parceling data, maintaining transmission

queues, and opportunistic forwarding during connectivity windows translate to P2P scenarios where aggregation nodes become neighbors in a mesh network.

2.2.3 Limitations of Centralized Coordination

The work on Resilient Edge-enabled IoT [8] addresses coordinator failures through dynamic leader election and backup mechanisms. Their coordination model divides environments into collaboration areas, with resource-rich edge devices serving as coordinators that allocate tasks to workers under their supervision when problems arise. Workers, in turn, are *active agents* such as robots and IoT devices that reside in a particular environment, detect problems, and notify their coordinators.

When coordinators fail, the system automatically elects backups through adaptive decentralized consensus, providing "gentle degradation" during failures with restoration after recovery. Multiple coordinators operate independently, eliminating single points of failure within the coordination model itself.

This architecture depends fundamentally on edge servers running JVM-based SCAFI middleware (a Scala library) to execute the aggregate programs that specify coordination behavior, and these heavyweight infrastructure requirements – both the Java runtime environment and resource-rich edge computing nodes – conflict with embedded platform constraints and infrastructure-failure scenarios.

queremos mingle com o Babel... se calhar não entrar por aqui ou tudo bem porque não vamos usar os big-raspis para coordination?

While the aggregate computing paradigm separates concerns (sensing, actuation, communication, coordination), the implementation assumptions make it unsuitable for disaster-resilient systems where edge servers may be the infrastructure that fails. The formal guarantees of self-stabilization and compositional properties come at the cost of persistent computational infrastructure that an embedded-focused deployment cannot provide.

2.2.4 Discussion

Existing peer discovery and topology mainteance approaches demonstrate mechanisms appropriate for distributed operation but not without their limitations for resource-constrained platforms aimed at disaster scenarios.

HyParView's [23] hybrid view architecture (small active, large passive) provides remarkable resilience to node failures, but the assumption of routing and TCP availability and persistent connections is not suitable for embedded devices using connectionless radio technology or operating under intermittent connectivity conditions.

The multi-protocol gateway's [21] BLE-based neighbor discovery works on our targeted hardware but imposes tree topologies with parent-child communication restrictions that limit route diversity and create dependency chains.

RPL routing in the heterogeneous disaster architecture [34] demonstrates practical WSN operation but has significant throughput limitations: 20-second minimum send intervals at 10 hops, and restriction of critical traffic to 1-hop from root nodes.

The store-and-forward mechanism presented in the elastic bandwidth framework [29] enables partition tolerance but targets cloud-backed deployments rather than P2P mesh scenarios.

Coordination frameworks like the one in [8] provide dynamic leader election but depend on JVM-based middleware on resource-rich edge servers.

We have found no work that integrates connectionless neighbor discovery, hybrid view maintenance, true P2P routing (not tree-based), and partition-tolerant store-and-forward on resource-constrained embedded platforms.

added narrative connector for next section

However, topology alone does not guarantee resilience, especially if all nodes still rely on a single form of communication, shifting the focus from network coordination to radio availability.

REMOVED: ubabel's approach will...

2.3 Multi-Radio Communication and Adaptive Protocol Selection

added narrative thingamagig

Even if devices can self-organize without fixed coordinators, the remaining issue becomes their ability to communicate across heterogeneous protocol stacks.

Single-radio (i.e. Wi-Fi only, BLE only, LoRa only, etc.) communication architectures are susceptible to a fundamental vulnerability: if their chosen medium becomes unavailable or inefficient due to interference, range limitations, or infrastructure failures, the entire system loses connectivity and most times ceases to operate altogether.

This becomes especially grave in disaster scenarios where communication conditions change unpredictably due to factors such as Radio frequency (RF) interference caused by debris, obstacles blocking line-of-sight propagation, or damaged/unavailable access points.

Existing approaches to multi-radio IoT systems – or multi-protocol, as commonly used in the literature – can be divided into three categories: multi-channel resilience architectures that orchestrate communication technologies for emergency scenarios; implementations that show the feasibility of radio heterogeneity on embedded platforms; and adaptive selection frameworks that switch protocols based on runtime conditions.

2.3.1 Multi-Channel Resilience Architectures

The Always Connected Things (AWCT) framework [24] orchestrates Low-Power Wide-Area Network (LPWAN) on top of LoRaWAN with ad-hoc networks (Bluetooth and Wi-Fi) specifically for standby emergency communication. Their architecture adds three modules to standard IoT devices (Raspberry Pi boards, in their test case): a battery module for power management, a power interrupt handler that triggers emergency mode when the power grid fails, and an ad-hoc bridge that forwards packets between the Bluetooth, Wi-Fi and LPWAN interfaces.

The system leverages dense IoT device deployment to provide emergency coverage, demonstrating that existing infrastructure can still serve a purpose during scenarios where the main power grid suffers issues. However, AWCT's reliance on centralized LoRaWAN gateways for Internet connectivity creates a single point of failure when those gateways become unreachable.

A more comprehensive heterogeneous approach is presented in [34], which integrates High frequency (HF) radio with Near Vertical Incidence Skywave (NVIS) technology, satellite links, WSNs, and DTN with mobile drones for disaster monitoring. Their system uses RPL [2] with three separate instances to differentiate traffic by priority: human data (voice/text via Bluetooth) receives the highest priority, followed by drone-collected data and finally sensor data.

The NVIS backhaul provides 250 km coverage radius without line-of-sight requirements, offering a cost-effective alternative to satellite communications.

While demonstrating successful real-world validation in Antarctica and urban deployments, the architecture's core NVIS topology with centralized coordination contrasts fully distributed operational requirements. Furthermore, their WSN layer requires a minimum 20-second sending interval to maintain acceptable packet loss rates at 10 hops, which highlights throughput limitations of single-channel tree topologies.

Security-focused multi-channel approaches like MCSC-WoT [6] combine Advanced Encryption Standard (AES) encryption with dynamic channel hopping across 2.4 GHz Wi-Fi channels to defend against jamming and eavesdropping attacks. Their lightweight synchronization mechanism minimizes energy consumption while maintaining security through Frequency-hopping spread spectrum (FHSS) patterns generated via Pseudorandom number generator (PRNG). Nodes that lose synchronization can rejoin by hopping to random channels and waiting for the next synchronization signal.

However, the system depends on a master node broadcasting synchronization signals and uses an initial PRNG seed shared among all nodes, raising questions about scalability and seed distribution mechanisms – particularly how new nodes can acquire seeds if deployed to a network some time after the system is operational.

2.3.2 Protocol Heterogeneity on Embedded Platforms

The work presented in [21] shows the technical feasibility of multi-radio operation on commodity Wi-Fi/BLE modules by integrating ESP-NOW, ZigBee, and Modbus protocols on devices from the ESP32 family to construct multi-hop, tree-based wireless networks.

FIXED: changed this to be more focused on the technical aspect

ESP-NOW requires pairing between peers based on their MAC addresses, thus some form of discovery is required after deployment. Their implementation uses BLE advertising beacons for neighbor discovery with Received signal strength indicator (RSSI) measurements (for distance estimation), computing parent selection priorities based on

weighted combinations of child count, RSSI values, and hop count in the network tree of a given node.

Testing demonstrated successful coexistence with ZigBee sensors, and supports the idea that different technologies can cooperate to overcome individual limitations.

2.3.3 Runtime Adaptive Protocol Selection

The MINOS platform [43] exemplifies the limitations of centralized multi-protocol approaches. While providing sophisticated multi-protocol support (CORAL-SDN and Adaptable-RPL) with dynamic protocol deployment and real-time parameter tuning, the system depends fundamentally on a centralized SDN controller, MQTT broker, and web server infrastructure. The architecture's single point of failure means that when the controller becomes unreachable the entire system loses its adaptive capabilities and reverts to static operation at best, or complete failure at worst.

FIXED: removed MDPI paper

This centralization pattern persists in other multi-protocol/multi-radio platforms despite their sophistication. MARS [38] provides Machine learning (ML)-based radio selection between ZigBee and BLE on Raspberry Pi devices, achieving an increase of up to 49% in throughput thanks to optimized switching, but requires fixed gateways for ML decision-making making it unsuitable for decentralized operation. The Chorus [32] routing framework also operates on Raspberry Pi devices and utilizes ZigBee, BLE and Wi-Fi, however, it places routing decisions on a cloud coordinator that floods the network *a priori*, meaning it cannot adapt to disaster circumstances should that coordinator fail.

2.3.4 Discussion

Existing frameworks demonstrate the technical feasibility of heterogeneous channel/protocol coordination but rely on centralized control mechanisms incompatible with disaster scenarios. AWCT [24] depends on centralized LoRaWAN gateways for Internet connectivity, the heterogeneous disaster architecture [34] uses centralized NVIS coordination with minimum 20-second send intervals at 10 hops, MCSC-WoT [6] requires master nodes broadcasting synchronization signals, the multi-protocol gateway [21] validates multi-radio operation but imposes tree topologies with gateway coordinators as single points of failure, and MINOS [43], MARS [38], and Chorus [32] all assume some form of persistent infrastructure.

We have found no work that addresses fully distributed multi-radio coordination where nodes can autonomously select and switch communication technologies without resorting to persistent infrastructure.

REMOVED: ubabel's approach will...

2.4 Decentralized Synchronization and Time Coordination

Multi-channel communication strategies, particularly those that employ coordinated channel hopping for security or efficiency reasons, require nodes to maintain synchronized clocks. Without time synchronization, devices cannot agree on when to switch channels, rendering coordinated multi-protocol operation unfeasible.

Traditional master-based synchronization approaches – where a designated coordinator is responsible for communicating timing signals – once more fall into the limitation of having a single point of failure in that same master node: should it fail, the entire network becomes susceptible to losing accurate temporal reference, leading to the collapse of any coordination efforts.

This is a notorious challenge when designing distributed systems, and it becomes more prominent in the disaster scenarios discussed thus far, where infrastructure failures are likely to eliminate the nodes responsible for time synchronization in a given system. Achieving robust time synchronization in such situations requires fully decentralized approaches that eliminate central coordinator dependencies while still remaining sufficiently lightweight to be executed on resource-constrained platforms.

2.4.1 Master-Based Synchronization as Counterexample

Time synchronization in distributed networks is frequently implemented through beacon or master-clock schemes due to their precision and implementation simplicity. As discussed in Section 2.3.1, the MCSC-WoT framework [6] demonstrates a security-focused multi-channel hopping approach aimed at embedded platforms, but with a fundamental dependency on a master node broadcasting synchronization signals.

This approach uses a shared PRNG seed to generate channel-hopping sequences, with the coordinator node broadcasting periodic synchronization beacons that participant nodes use to compensate for clock drift. Nodes that lose synchronization altogether can rejoin by hopping to a random channel and waiting for the next coordinator beacon.

While the proposed goals were achieved in terms of minimizing energy expenditure and maintaining FHSS patterns, it inherits the fundamental limitation of all master-based approaches mentioned at the start of Section 2.4. If the master fails or becomes unreachable, participating nodes gradually drift out of synchronization until coordinated channel hopping is no longer possible.

The system's clock drift compensation algorithm demonstrates the feasibility of synchronization on ESP32 platforms, and their measured AES encryption performance validates that lightweight security can coexist with time synchronization on resource-constrained devices. However, the architectural dependency on a central coordinator fundamentally conflicts with infrastructure-independent operation requirements we aim to fulfill.

2.4.2 Gossip-Based Masterless Synchronization

Decentralized time synchronization eliminates coordinator dependencies by having nodes reach consensus on clock values through distributed local interactions. Two complementary approaches demonstrate the viability of gossip-based synchronization for WSNs.

The Randomized gossip-consensus-based sync (RGCS) algorithm proposed in [48] addresses time synchronization in dynamic WSNs through randomized asynchronous gossip. Each node maintains a logical clock (T) composed of rate (α) and offset (β) parameters, which together transform the node's hardware clock (τ) into synchronized logical time.

Rather than requiring fixed communication links between specific node pairs which might be fragile in dynamic topologies, their approach uses Poisson-based randomized link activation where each potential synchronization link activates with intensity λ.

The synchronization process operates through pairwise gossip exchanges: when a link activates, the triggering node sends a Sync-L beacon selecting a triggered neighbor, followed by bidirectional exchange of multivariable messages containing each node's current logical clock parameters $[\alpha, \beta, \tau]$.

The asynchronous randomized timing of these exchanges is advantageous in that collision rates drop to near zero compared to 19-23% for deterministic communication protocols, as independent Poisson intervals make simultaneous transmissions to the same receiver statistically unlikely.

The proposed RGCS employs a converge-to-max criterion rather than average-value consensus. This maximum-based approach achieves finite time convergence significantly faster than average-based protocols that require many iterations to converge under significant clock drift. Offset compensation follows suit, with nodes adjusting β parameters based on the difference between their and the neighbors' logical clocks.

Bounded communication delays are handled through a least-square estimation low-pass filter.This addresses the realistic concern of uplink and downlink delays differing, and avoids the symmetric delay assumptions made by many theoretical protocols. The filter's weighing parameter decreases over time, in order to restrain the negative effects of additive noise in stochastic approximation.

Storage complexity remains $O(|N_i|)$ per node per iteration (proportional only to the number of neighbors, not network size) thus ensuring scalability. The protocol simultaneously compensates both clock rate and offset, unlike approaches that handle these separately and thus require additional convergence time.

2.4.3 Coordination Through Passive View Maintenance

As discussed in Section 2.2.1, the shuffle-based passive view maintenance presented in HyParView [23] provides a complementary coordination mechanism. While primarily designed for topology management, the periodic shuffle operations in which nodes

exchange lists of known peers can also provide a *catalog* of potential synchronization partners beyond their immediate active neighbors.

FIXED: moved NTP and GPS talk to discussion

Nodes performing shuffle exchanges already communicate periodically; these same communication windows can opportunistically carry synchronization messages (piggybacking), reducing protocol overhead. The passive view serves as a pool of potential sync partners, so that when a node's active sync neighbors become unreachable, it can initiate sync exchanges with passive view members, providing added resilience to topology changes without requiring global network knowledge.

2.4.4 Discussion

Coordinator-based synchronization approaches like MCSC-WoT [6] demonstrate feasibility on ESP32 platforms with measured AES encryption coexistence, but create single points of failure when master nodes become unreachable.

RGCS [48] eliminates coordinator dependencies, and provides converge-to-max synchronization with Poisson-based randomized gossip that achieves near-zero collision rates with just $O(|N_i|)$ storage complexity per node. However, it assumes homogeneous single-protocol networks where all nodes use the same communication medium with identical energy characteristics and range properties.

HyParView's [23] shuffle-based passive view maintenance could provide complementary coordination through periodic peer exchanges carrying piggybacked sync messages, but was not designed for time synchronization. One example would be when a node receives a shuffle message containing peer descriptors (IDs and capabilities): it can evaluate these peers as candidates for time synchronization based on their advertised characteristics – for instance, prioritizing peers with access to Network Time Protocol (NTP) or Global Positioning System (GPS) clock synchronization.

We have found no work that addresses decentralized time synchronization across heterogeneous multi-protocol networks where different communication technologies (BLE, LoRa, Wi-Fi) have distinct energy costs, range, and reliability properties that warrant protocol-specific gossip rates.

REMOVED: ubabel's approach will...

2.5 Security Considerations and Scope Delimitation

FIXED: delimitar scope, still mention que olhei para security stuff

This thesis focuses on resilience and decentralization at the physical and link layers, and, as such, security mechanisms operating at higher layers (i.e., authentication, key management, privacy-preserving mechanisms) are considered out of scope for the present body of work.

Security and privacy are nonetheless critical concerns in emergency and disaster-focused scenarios, where sensitive data such as location tracking and environmental

conditions could be the target of malicious actors seeking to disrupt system operation, inject false information, or compromise user safety.

Substantial research effort has been devoted to addressing these concerns in resource-constrained IoT systems. While a comprehensive discussion of such mechanisms is beyond the scope of this work, a brief overview of relevant approaches is provided below to contextualize potential future extensions of the proposed μ**-Babel** framework.

Several lightweight authentication and key management approaches demonstrate that cryptographic operations based on hash functions, symmetric encryption and polynomial secret sharing are feasible on embedded platforms. The Group Authentication Scheme at the Edge (GASE) [31] relies on aggregated Message Authentication Codes (MACs) and threshold techniques to reduce computational overhead, while the Lightweight Polynomial-based Key Management (LPKM) protocol [15] enables scalable and pairwise key establishment with low storage requirements and distributed revocation. Symmetric encryption – notably AES – has been proven to be practical on ESP32 devices without prohibitive performance or energy costs, as demonstrated by the MCSC-WoT framework [6].

Other lines of work approach privacy-preservation by relying on hardware fingerprints and pseudonym systems, including protocols based on Physically Unclonable Functions (PUFs) [18, 49, 10]. However, these approaches assume specialized hardware or centralized verification infrastructure, greatly limiting their applicability to commodity devices and infrastructure-less deployments. Similarly, zero-preloading key agreement schemes [25] attempt to eliminate prior trust establishment but face unresolved challenges related to authentication bootstrapping and revocation.

Several proposals focus instead on blockchain technology or distributed ledgers to address IoT security concerns, including access control [44] and decentralized Public key infrastructures (PKIs) [46]. While these approaches remove single points of failure, they assume persistent connectivity incompatible with disaster scenarios and computational capabilities far beyond those available in low-end embedded platforms; pairing-based decentralized encryption schemes further exacerbate this issue [20].

Ultimately, existing work confirms the feasibility of lightweight security mechanisms on constrained devices, but also highlights a persistent reliance on infrastructure availability or pre-established trust. Rather than addressing these challenges directly, the proposed μ**-Babel** framework focuses on providing a foundation for resilient and decentralized communication on top of which security mechanisms may be integrated.

2.6 Summary

Table 2.2 summarizes the key architectural differences between existing work and **μ-Babel**'s planned vision across the technical domains discussed in this chapter.

Table 2.2: Related work summary

Domain	**Related Work Limitations**	**μ-Babel** Approach
Radio coordination	Master nodes (MCSC-WoT), SDN controllers (MINOS), gateway coordinators (multi-protocol gateway)	Fully distributed selection, adaptation based on application context
Mesh topology	Tree-based restrictions, TCP-dependent links (HyParView), centralized coordination	Connectionless BLE/LoRa discovery, hybrid views, true P2P routing
Time sync	Master beacons (MCSC-WoT), single-protocol gossip (RGCS)	Protocol-specific Poisson rates, converge-to-max

FIXED: removed seurity row

A common pattern is present across these domains: existing approaches rely on centralized coordination mechanisms that become single points of failure during infrastructure collapse.

Protocol coordination depends on master nodes or SDN controllers, mesh topologies assume consistent connectivity or gateway coordinators, time synchronization requires master beacons.

Even distributed approaches like HyParView and RGCS make assumptions (routing infrastructure, persistent TCP connections, single-protocol homogeneity) incompatible with resource-constrained multi-protocol disaster scenarios.

In the next Chapter we discuss how these limitations inform the design of **μ-Babel** as motivating factors for developing a fully decentralized architecture – one that prioritizes autonomous operation, protocol diversity, and resilience under unstable connectivity.

FIXED: closing paragraph, mais narrativa etc?

There we highlight the system requirements, operating conditions, and architectural choices that will enable robust communication and coordination across heterogeneous devices.

3

Solution Architecture

3.1 Recap

3.1.1 The Problem

Current Internet of Things (IoT) and domotics systems assume continuous infrastructure availability, an assumption that proves rather catastrophic during disasters, in hazardous environments, or when privacy requirements demand local autonomy (Chapter 1).

The widely-adopted, cloud-centric architectural model creates single points of failure in myriad ways: devices depend on remote servers for control logic, local access points for Internet connectivity, and designated coordinators for synchronization.

When infrastructure fails, whether due to large-scale outages [42] or disaster-related network collapse, coordinated operation becomes impossible despite the local availability of physical computational resources.

3.1.2 Technical Challenges

Throughout Chapter 2 we identified the core challenges that prevent existing approaches from fully addressing the overarching problem of reliable autonomous operation:

Centralized Coordination: Radio selection requires master nodes or Software-defined networking (SDN) controllers. Topology management depends on gateway coordinators. Time synchronization relies on master beacons. Authentication requires persistent servers or group leaders. When these central elements fail or become unreachable, coordination collapses despite local devices remaining operational.

Single-Radio Operation: Gossip-based synchronization assumes homogeneous networks where all nodes share identical communication characteristics, and topology maintenance protocols assume either persistent connections or uniform radio properties ill-suited for embedded platforms with access to multiple wireless technologies.

Infrastructure dependencies: Blockchain-based security requires connectivity to distributed ledgers. Edge computing assumes eventual cloud access. Hybrid architectures distribute processing but remain dependent on infrastructure for coordination. Systems designed with "infrastructure as fallback" still fail when that fallback becomes unavailable indefinitely.

Resource constraints: Sophisticated coordination mechanisms exceed embedded platform capabilities. Security schemes require specialized hardware (Physically Unclonable Functions (PUFs)) or computationally expensive operations (bilinear pairings). The tension between autonomous operation demands and limited computational, memory, and energy budgets remains unresolved in existing approaches.

μ-Babel will address these interconnected challenges through a unified framework designed specifically for resource-constrained multi-radio operation during infrastructure failures. The following sections detail the design goals, targeted platforms, and design decisions that enable autonomous coordination without central dependencies.

3.2 System Requirements

3.2.1 Overview of μ-Babel's Approach

In order to effectively overcome the difficulties exposed in the previous chapter, we can highlight a set of Functional Requirements (FRs) that **μ-Babel** must meet in order to eliminate the *dependency* on centralized coordination, continuous connectivity, and cloud infrastructure; and a set of Non-Functional Requirements (NFRs) which would strengthen the overall quality of the framework and ensure it can be widely adopted with ease.

Functional Requirements

Infrastructure Independence: The system must maintain communication capabilities when traditional infrastructure becomes unavailable. Rather than treating this unavailability as an exceptional condition requiring failover mechanisms, our architecture will focus on autonomous Peer-to-peer (P2P) connectivity as the baseline mode of operation, with infrastructure integration as an added benefit when available, rather than a hard dependency.

Autonomous Operation: Devices will self-organize into functional networks, dispensing centralized coordinator nodes. Instead, local P2P gossip interactions will be the basis for coordination and synchronization.

Multi-Radio Exploitation: Communication strategies will adapt to the available mediums and device resources without centralized control. Communications will continue

despite disruption of individual radio interfaces through adaptive selection and opportunistic switching. Interface diversity provides resilience through redundancy, and this should be achieved with minimal impact on the upper layers of the application (through abstractions).

Scalability: The system must support deployments ranging from small, localized networks to large-scale, highly distributed collections. Rather than requiring global knowledge of all participants, nodes organize into local groups where elected leaders maintain state only for their immediate members. This area-based approach – analogous to Open Shortest Path First (OSPF)'s autonomous systems [30] – distributes coordination load and enables networks to scale while keeping per-node overhead bounded by local group size.

Resource Efficiency: Embedded platforms inevitably present strict resource constraints (i.e., memory storage, processing, battery), thus all protocols and algorithms must execute within these limitations. In order to enable this, adaptive selection of the employed radio technologies will take into account device capabilities and available resources.

Non-Functional Requirements

"something something these are not essential ..." ou o parágrafo inicial basta?

Optional Cloud Integration: When infrastructure *is* available, it should be taken advantage of. Thus, **μ-Babel** will exploit it for tasks that are beyond the capabilities of individual nodes or even gateways, such as data aggregation, analytics, and long-term storage. Cloud integration thus becomes an optional – and welcome – addition to the repertoire of features, but does not itself become a requirement for dependable operation.

Interoperability: Following optional cloud integration, widely used IoT protocols such as MQTT and Constrained Application Protocol (CoAP) can be leveraged to enable integration with existing IoT platforms and services when conditions allow.

3.3 Operating Conditions and Failure Modes

μ-Babel is designed to remain operational under conditions where traditional infrastructure becomes unreliable or unavailable. This section establishes the operational environment and failure scenarios that drive our architectural decisions.

3.3.1 Target Operating Environments

μ-Babel targets deployments where infrastructure availability cannot be guaranteed, and in which autonomous operation is a necessity:

Disaster Scenarios: Locations susceptible to natural disasters (i.e., earthquakes, floods) or infrastructure failures that eliminate power grids and network access points.

Remote or Hazardous Locations: Industrial sites, wilderness monitoring, or other dangerous environments where manual intervention and infrastructure maintenance are impractical or impossible.

Local-First Deployments: Environments where data sovereignty requirements or privacy concerns require local processing and control rather than cloud dependency, even when connectivity is available.

add more focus on smart home stuff?

3.3.2 Expected Failure Modes

The architecture must maintain functionality under the following conditions:

Infrastructure Collapse: Complete loss of access points and external network connectivity. Devices must self-organize and maintain P2P communication without infrastructure support.

Communication Medium Disruption: Individual radio interfaces may become unavailable due to interference, jamming, congestion, or physical obstacles blocking signal propagation. Technology diversity provides alternative communication paths and ensures continued operation.

Network Partitioning: Physical damage or communication range limitations may split networks into isolated subgroups. Devices within partitions must continue autonomous operation while supporting eventual reconciliation when connectivity is restored.

Node Failures: Individual devices may become unavailable due to battery depletion, physical damage, or environmental conditions. The network must maintain operation despite individual node failures without depending on specific coordinator nodes.

Intermittent Connectivity: Communication links may be unstable with varying packet loss, latency, and availability. Protocols must tolerate unreliable links and adapt to changing conditions.

3.4 Hardware Platform

μ-Babel targets resource-constrained embedded platforms commonly used in IoT and domotics systems. The following sections cover device specifications and radio technology characteristics, establishing the foundation for our hybrid architecture.

We distinguish between two device classes based on their capabilities and roles within the architecture.

3.4.1 Class-1: Battery-Powered Sensor Nodes

The primary targets for **μ-Babel**. These embedded devices handle sensing, actuation, P2P communication, and decentralized coordination without requiring external infrastructure.

Table 3.1 presents the resources available across Class-1 devices. These span from basic sensor nodes (Pico 2W, ESP32-C5/C6) to more capable embedded devices (ESP32-S3/P4), enabling role differentiation within the autonomous peer network.

TODO: figure out table spacing

Table 3.1: Class-1 Device Specifications

Device	CPU[1]	CPU Clock	SRAM	PSRAM[2]	Flash
Raspberry Pi Pico 2 W	Dual-Core Arm Cortex-M33	150 MHz	520 KB	✗	4 MB
ESP32	Dual-Core Xtensa LX6	240 MHz	520 KB	✗	16 MB
ESP32-C5	Single-Core RISC-V	240 MHz	384 KB	✗	4 MB
ESP32-C6	Single-Core RISC-V	160 MHz	512 KB	✗	4 MB
ESP32-S3	Dual-Core Xtensa LX7	240 MHz	512 KB	8 MB	16 MB
ESP32-P4	Dual-Core RISC-V	360 MHz	768 KB	32 MB	32 MB

[1] All supported devices are 32-bit
[2] Pseudostatic random-access memory (PSRAM) is supported on all devices, but not available in the particular models used during development

Resource availability within Class-1 devices varies significantly both in processing power and available memory. While there are no fixed roles, capability awareness naturally influences dynamic selection procedures:

Data Aggregation and Buffering: Higher-resource devices (ESP32-S3/P4) can maintain larger message buffers and aggregate data from multiple sensor nodes, which is particularly valuable during network partitions when store-and-forward mechanisms come into effect.

Interface Bridging: Devices with broader communication technology support naturally serve as bridges between domains, forwarding messages across different mediums based on advertised capabilities.

Radio Interface Diversity and Device Coverage

Table 3.2 shows the different radio technologies supported across Class-1 devices, demonstrating considerable heterogeneity.

The wide variety of communication support across devices provides several advantages:

Coverage: All devices support at least three radio communication technologies (Wi-Fi, Bluetooth Low Energy (BLE), LoRa), ensuring baseline hybrid operation across the entire platform range, such that even the least capable device (Pico 2 W) can participate in

Table 3.2: Supported Wireless Technologies

Device	Wi-Fi	Bluetooth	ESP-NOW	ZigBee	LoRa[1]
Pico 2W	2.4GHz	Classic + Low Energy (LE) 5.2	✗	✗	✓
ESP32	2.4 GHz	Classic + LE 4.2	✓	✗	✓
ESP32-C5[2]	2.4/5 GHz	LE 6.0	✓	3.0	✓
ESP32-C6[2]	2.4 GHz	LE 5.3	✓	3.0	✓
ESP32-S3	2.4 GHz	LE 5.0	✓	✗	✓
ESP32-P4[3]	2.4 GHz	LE 5.3	✓	✗	✓

[1] Via external SX1276/SX1262 transceiver, controlled through Serial Peripheral Interface (SPI)
[2] ESP32-C5/C6 also support 802.11ax (Wi-Fi 6) and Thread, but we focus on ZigBee given its wider adoption in IoT systems
[3] ESP32-P4 does not possess wireless capabilities by itself and relies on a C6-Mini coprocessor, with ZigBee support still in development

heterogeneous mesh networks. Particularly, LoRa is supported using external transceivers via SPI, ensuring long-range connectivity across the board.

Device-Specific Advantages: ESP32 family devices can leverage ESP-NOW for P2P communication without infrastructure. ESP32-C5/C6 additionally provide ZigBee support for standardized mesh networking – particularly interesting in building automation and domotics contexts.

Notably, ESP32-C5 supports ESP-NOW over both 2.4 and 5 GHz bands, which provides a crucial short-range alternative path for congested environments where BLE, Wi-Fi and ZigBee all contend for 2.4 GHz channels.

These factors combine to provide remarkable resilience: spectrum diversity across 2.4/5 GHz and sub-GHz bands, infrastructure-free operation via ESP-NOW and BLE, and long-range connectivity through LoRa.

Together, these Class-1 specifications and radio capabilities establish the core hardware foundation for **μ-Babel**'s autonomous operation. Section 3.6 details how these resources are exploited in the proposed architecture.

3.4.2 Class-2: Optional Aggregation Nodes

Resource-rich devices that run the broader Babel [16] framework and are mentioned here for the sake of integration with broader systems, but are not required for **μ-Babel**'s core autonomous operation.

This class encompasses general-purpose computing devices such as desktops, laptops, and servers. For development purposes, and staying within the realm of embedded platforms, we specifically utilize Raspberry Pi Single-board computers (SBCs), whose specifications are available in Table 3.3.

Class-2 devices serve as *optional* infrastructure that, when present, can perform: (1) Data aggregation and long-term storage; (2) Computationally intensive analytics beyond

Table 3.3: Class-2 Device Specifications

Device	CPU[1]	CPU Clock	RAM[1]	Storage
Raspberry Pi 5	Quad-Core 64-bit Arm Cortex-A76	2.4 GHz	4-16 GB	Variable external
Raspberry Pi 4	Quad-Core 64-bit Arm Cortex-A72	1.8 GHz	4-8 GB	Variable external

[1] Configurations with less Random-access memory (RAM) are available but were not considered

Class-1 capabilities; and (3) Optional cloud bridging when external connectivity is available and desired (e.g., MQTT brokerage, CoAP endpoints).

Critically, these devices *do not* serve as coordinators or single points of failure. **μ-Babel** networks will operate fully autonomously with Class-1 devices alone; Class-2 nodes simply participate as peers that happen to have greater resources. Their absence does not prevent network formation, peer discovery, or autonomous operation.

3.5 Software Stack and Development Environment

μ-Babel is implemented on top of well-established embedded software platforms that provide hardware abstraction and concurrency support while remaining sufficiently lightweight for resource-constrained platforms.

The system leverages vendor-supported Software Development Kits (SDKs) and a minimal Real-time operating system (RTOS) to ensure portability across Class-1 devices while maintaining fine-grain control over hardware resources.

Operating System Layer FreeRTOS [4] serves as the RTOS across all Class-1 devices. It provides deterministic task scheduling and inter-task communication implemented around a single queue primitive [5]. This queue mechanism forms the basis for queues (as data structures), semaphores, and mutexes. FreeRTOS also allows for configurable memory allocation and per-task stack management, all without the overhead of a full-featured Operating system (OS).

This allows multiple protocol components and I/O handlers to execute concurrently while preserving predictable timing behavior, which is essential for networking and radio-driver applications.

FreeRTOS is the *de facto* OS on ESP32-family devices, and is easily brought into the Pico platform. By using it as a common execution runtime, **μ-Babel** maintains a consistent concurrency and timing model across heterogeneous microcontroller platforms, simplifying both portability and implementation.

Platform SDKs and Hardware Abstractions Hardware-specific functionality is accessed through vendor abstractions. The Espressif IoT Development Framework (ESP-IDF) [13] is used for the ESP32 family, while the Raspberry Pi Pico SDK [35] is used for RP2350-based devices.

These frameworks provide Application programming interfaces (APIs) for peripheral access (e.g., General-purpose input/output (GPIO), SPI, Inter-Integrated Circuit (I^2C), Universal asynchronous receiver-transmitter (UART)), timers, interrupt handling, Wi-Fi/BLE radio stacks, and – where available – hardware-accelerated cryptographic operations.

By encapsulating platform-specific bare-metal details behind stable interfaces, they allow **μ-Babel** to target multiple microcontrollers through a set of thin Hardware Abstraction Layers (HALs), enabling re-utilization of higher-level protocols and coordination logic across Class-1 devices. **μ-Babel** is implemented in C, as it allows precise control over memory usage, interrupt behavior, and timing, which are essential in embedded environments with limited memory and strict real-time constraints.

Concurrency Model **μ-Babel** follows a task-based concurrency model built on FreeRTOS primitives. Protocol handlers execute as independent tasks, allowing computation, I/O, and radio operations to execute concurrently and predictably. Inter-task communication is handled through message queues, while mutexes and semaphores are used to protect shared state and coordinate access to hardware resources.

3.6 System Model

μ-Babel is structured as a layered architecture that provides foundational services for distributed and autonomous communication using heterogeneous radio/wireless technologies. We distinguish between *core components*, and *protocols*, where a protocol is a state machine implementing a distributed algorithm, coordination mechanism, or application-specific logic.

Figure 3.1 provides an overview of the system architecture, with the core components that form the basis of the proposed framework, and two complementary protocols geared towards autonomous peer discovery and resilient topology management, explained in more detail below.

3.6.1 Core Components

These provide the base runtime upon which all protocols operate. They handle protocol registration and lifecycle management, event routing, wireless communication abstractions, and time synchronization.

Protocol Manager Maintains a registry of active protocols and manages their lifecycle throughout system operation, handling initialization, resource allocation, and shutdown.

Key responsibilities include: (1) Registering protocols and their associated event queues during initialization; (2) Creating and managing RTOS tasks for execution (one task per protocol); (3) Tracking protocol state (running, suspended, stopped); (4) Loading protocol

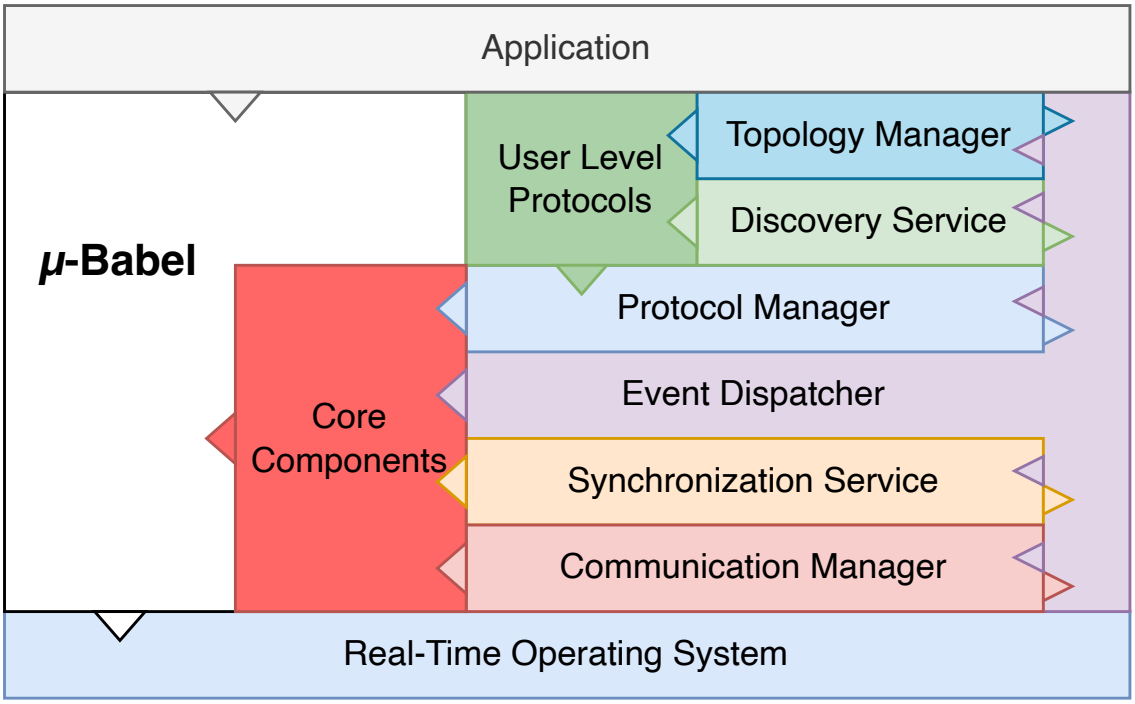

Figure 3.1: System architecture overview

configurations (i.e., from an SD card); (5) Allocating and monitoring protocol resources; (6) Providing protocol lookup services for directed message routing.

Event Dispatcher Implements an event-driven coordination model that decouples components and enables asynchronous communication between protocols and services. The dispatcher supports direct messaging between protocols and publish/subscribe delivery.

Main responsibilities include: (1) Managing component subscriptions to specific event types (i.e., notifications, timers, messages) and subtypes (protocol-specific); (2) Broadcasting events to all registered subscribers for publish/subscribe style communication; (3) Routing directed messages to specific protocols using protocol identifiers for point-to-point communication; (4) Managing event memory lifecycle through reference counting as events are delivered to multiple subscribers.

This enables system-wide notifications and targeted inter-protocol messages while preventing tight coupling between components and application logic.

Communication Manager Manages the hybrid wireless capabilities of the device, handling multiple communication technologies (Wi-Fi, BLE, LoRa, etc.) and providing agnostic abstractions to higher layers.

Its main responsibilities include: (1) Establishing and maintaining (virtual) connections to remote peers identified by Universally unique identifiers (UUIDs); (2) Detecting connectivity events (i.e., new connections, disconnections) and notifying interested components via the Event Dispatcher; (3) Transmitting outbound messages (via unicast, multicast,

or broadcast) over appropriate wireless channels; (4) Receiving incoming messages and delivering them to the protocol layer via the Event Dispatcher.

This component provides a unified interface for on-device protocols, allowing higher layers to interact with mesh network participants via their UUID without concern for the underlying wireless technologies involved.

Time Synchronization Service Maintains a logical clock composed of rate and offset parameters that transform a device's hardware clock into synchronized logical time, implementing decentralized consensus on clock values through randomized gossip, inspired by Randomized gossip-consensus-based sync (RGCS) [48].

Using Poisson-distributed intervals to minimize collision probability, it randomly selects sync partners from active connections maintained by the Communication Manager. Sync messages can be piggybacked on outgoing protocol messages to reduce communication overhead.

Key aspects include: (1) Technology-specific Poisson gossip rates accounting for energy costs and reliability (e.g., different rates for BLE vs. LoRa); (2) Converge-to-max criterion for faster convergence than average-based approaches; (3) Simultaneous rate and offset compensation;

Unlike coordinator-centric synchronization that creates single points of failure, this gossip-based approach maintains synchronization through distributed local interactions.

3.6.2 User-Level Protocols

A user is, in effect, an application developer utilizing the core components of μ**-Babel**. Thus, we will provide two protocols out-of-the-box that will take full advantage of and enhance the core components: one for peer discovery, and another for topology management, which, when combined, enable autonomous mesh operation without reliance on centralized coordination.

Peer Discovery Service Implements neighbor discovery across different wireless technologies. Short-range radios like BLE, ESP-NOW, and ZigBee employ active discovery mechanisms (i.e., advertising, peer lists, beacons), while long-range LoRa communication relies on opportunistic message exchange.

The service discovers peers and provides standardized information to the Topology Manager by: (1) Periodically scanning for neighbor advertisements and beacons; (2) Advertising local device capabilities (supported radio interfaces, computational resources, battery status); (3) Translating technology-specific discovery information into standardized peer descriptors for the Topology Manager.

Topology Manager Provides resilient overlay membership by maintaining hybrid partial views (HyParView [23]) of a network of devices, enabling the mesh to survive infrastructure

failures. It has two modes of operation: when infrastructure is available (e.g., Wi-Fi access points), it builds and maintains an overlay topology that may span beyond direct neighbors; when infrastructure fails, the topology gracefully degrades to proximity-based membership using direct wireless links discovered by the Peer Discovery Service.

It maintains a small *active view* of peers for active communication, and a larger *passive view* of potential neighbors. The *active view* is managed reactively, such that nodes are added during join operations and removed upon failure detection; and the *passive view* is maintained through periodic shuffle operations that exchange peer descriptors with neighbors.

Key operations include: (1) Building overlay topology leveraging available infrastructure; (2) Periodic shuffling of passive view to maintain broader connectivity; (3) Graceful degradation to proximity-based topology upon infrastructure failure; (4) Automatic peer promotion from passive to active view for failover.

This protocol can serve multiple purposes: other protocols can query the active view to select communication partners (e.g., the Synchronization Service selecting gossip targets), the Communication Manager can prioritize maintaining connections to active view members, and the passive view provides automatic failover candidates when connections falter.

4

Progress Report and Planning

This chapter outlines the current status of the proposed implementation, the validation strategy, and development timeline for **μ-Babel**.

4.1 Progress Report

The implementation of **μ-Babel** is currently in its early stages, with protocol-specific handlers under development.

Hardware Abstraction Layers (HALs) for peripheral access (General-purpose input/output (GPIO), Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I^2C)) are functional across both ESP-IDF and Pico SDK platforms. The implementation of the Core Components (Section 3.6.1) is partially complete: the Protocol Manager supports basic protocol registration and lookup, the Event Dispatcher is operational, providing event-driven communication between components, the Communication Manager currently supports Wi-Fi peer connection management, and unicast/multicast message delivery. The Time Synchronization Service is still in the planning stage.

The User-Level protocols (Section 3.6.2) has one working component in the Peer Discovery Service, which currently supports discovery over Wi-Fi, while the Topology Manager is still in the planning stage.

Proof-of-Concept Deployment: The current system has been deployed and tested on M5Stack Core Basic [26] devices (ESP32-based) by commanding multiple Raspberry Pi 5 boards attached to actuators, demonstrating an "inversion of control" to signal the flexibility and potential of the proposed system.

4.2 Planning

4.2.1 Validation

To validate **μ-Babel**'s implementation, we intend to follow a three-tier approach with individual component verification, integration testing, and deployment evaluation, measuring

for different metrics at each turn.

Component Verification: (1) Message delivery rates, latency, and energy consumption per radio communication protocol (**communication manager**); (2) Discovery time and neighbor view consistency across radio combinations (**peer discovery**); (3) Active/passive view maintenance and failure detection latency (**topology manager**); (4) Clock drift compensation, gossip convergence speed, and multi-radio sync overhead (**synchronization service**).

Integration Testing: (1) Network formation time without access points or coordinators (**infrastructure independence**); (2) Switching latency and stability under changing conditions (**radio protocol adaptation**); (3) Recovery time after node failures, message delivery during network partitions (**fault tolerance**); (4) Performance degradation as network size increases (**scalability**); (5) Message delivery times across multi-hop paths with varying radio combinations (**end-to-end latency**);

4.2.2 Use Cases

To guide development and further validate μ**-Babel**'s implementation, we consider two complementary scenarios that showcase different aspects of the system's capabilities. Both use cases adhere to a general hierarchy:

Sensing Tier: Devices performing direct environmental or physical monitoring, typically using short-range wireless technologies for communication among themselves.

Actuation Tier: Devices providing physical feedback to users or the environment (i.e., displays, alarms, controls), receiving commands from higher tiers.

Aggregation Tier: Devices responsible zones or regions, aggregating data from the sensing tier, controlling the actuation tier, and bridging autonomous regions for widespread communication.

Coordination Tier: System-wide coordination and optional external integration (cloud services, building management systems), typically requiring more capable hardware.

These following scenarios are meant to demonstrate operation during normal and degraded conditions, taking advantage of a heterogeneous collection of devices with optional cloud integration when available.

Campus-Wide Environmental Monitoring

Deployment Context: Sensors distributed across multiple buildings in a university campus to monitor environmental conditions (e.g., temperature, humidity, CO_2 levels, occupancy) and classroom occupancy:

TODO: look into Millimeter Wave Radar

- **Sensing:** Battery-powered ESP32/Pico 2 W devices (Class-1) throughout rooms and corridors
- **Actuation:** Small displays showing current conditions and occupancy status; battery-power as backup if using low-power e-paper technology
- **Aggregation:** ESP32-S3/P4 devices (Class-1) as floor/building gateways using Bluetooth Low Energy (BLE)/ESP-NOW locally, and LoRa for inter-building communication; larger batteries supply this tier once grid power becomes unavailable
- **Coordination:** Grid-powered Raspberry Pi devices (Class-2) in server rooms, providing MQTT brokerage and cloud connectivity

Normal Operation Mode: Full hierarchical connectivity: sensing tier using BLE/ESP-NOW within rooms and corridors to gather environmental information; actuation tier using the same technologies to provide real-time data in information hubs; aggregation tier providing inter-building communication via LoRa; and coordination tier providing cloud integration.

This mode validates: (1) Hybrid wireless communication; (2) Topology management; (4) Optional cloud integration (Non-Functional Requirement (NFR), Section 3.2.1).

Offline Operation Mode: Upon infrastructure failure (simulated power outage or network collapse), coordination tier becomes unavailable but the other three continue autonomous operation through Peer-to-peer (P2P) gossip, with data buffered for later storage once regular connectivity is reinstated.

This mode validates: (1) Infrastructure independence (Functional Requirement (FR), Section 3.2.1); (2) Autonomous operation; (3) Communication technology adaptation; (4) Store-and-forward mechanisms; (5) Graceful degradation.

Building Safety and Evacuation System

Deployment Context: Safety monitoring system combining hazard detection (smoke, fire, CO_2) with evacuation guidance:

- **Sensing:** Class-1 hazard sensors at strategic locations (stairwells, corridors, rooms)
- **Actuation:** Class-1 devices with e-paper displays showing evacuation routes and hazard warnings
- **Aggregation:** Class-1 per-floor gateways maintaining floor-level coordination and dynamically computing safe routes
- **Coordination:** Class-2 building coordinator integrating with building management systems

Normal Operation Mode: Continuous monitoring with sensing tier using BLE/ESP-NOW to aggregation tier, which forwards to coordination tier for logging and analysis.

This mode validates: (1) Priority-based queuing (critical vs. periodic data); (2) Low-latency local communication for real-time monitoring; (3) Integration with existing infrastructure.

Disaster Operation Mode: Upon hazard detection or infrastructure failure, sensing tier broadcasts alerts via available wireless channels, aggregation tier autonomously computes evacuation routes and updates actuation tier displays which show real-time guidance. Critical functions remain operational without coordination tier.

layout info stored in an SD card? much to consider

This mode validates: (1) Autonomous operation; (2) Multi-radio redundancy for resilience; (3) Decentralized coordination; (4) Adaptive communication technology selections; (5) Real-time responsiveness.

4.2.3 Scheduling

In order to achieve the proposed requirements (Section 3.2.1), we delineate a set of tasks to be carried out in the coming months:

Task 1 – Developing Core Components: Protocol Manager, Event Dispatcher validation, Communication Manager, and Time Synchronization Service as the foundation of our framework.

Task 2 – Developing User-Level Protocols: Peer Discovery Service and Topology Manager for autonomous mesh operation.

Task 3 – Validating the implementation: Use case development parallel to Tasks 1 and 2, followed by experimental setup and performance evaluation.

Task 4 – Completing the Dissertation: Final document preparation for potential conference submission (PerCom 2027).

The GANTT chart in Figure 4.1 presents the timeline for completing these tasks.

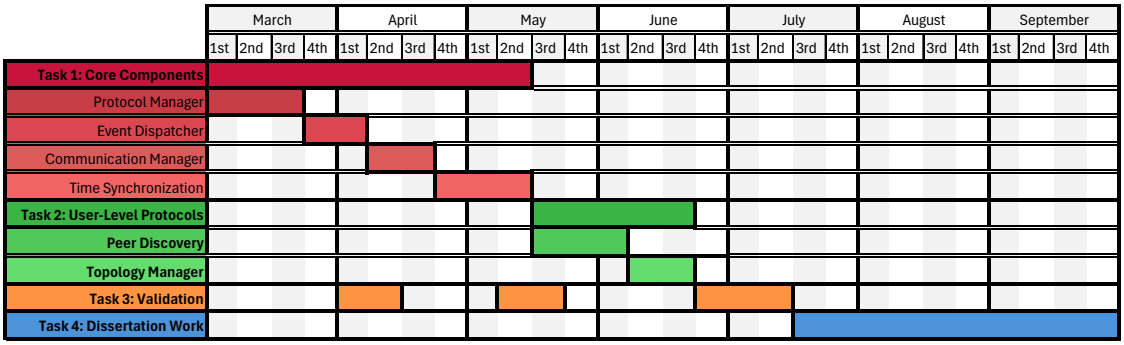

Figure 4.1: Proposed timeline

Bibliography

[1] S. M. Alamouti, F. Arjomandi, and M. Burger. "Hybrid Edge Cloud: A Pragmatic Approach for Decentralized Cloud Computing". In: *IEEE Communications Magazine* 60.9 (2022), pp. 16–29. DOI: 10.1109/MCOM.001.2200251 (cit. on p. 1).

[2] R. Alexander et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550. 2012-03. DOI: 10.17487/RFC6550 (cit. on p. 14).

[3] Z. Amiri et al. "Resilient and dependability management in distributed environments: A systematic and comprehensive literature review". In: *Cluster Computing* 26.2 (2023), pp. 1565–1600 (cit. on p. 2).

[4] AWS open source. *FreeRTOS*. 2024. URL: https://www.freertos.org/ (visited on 2026-01-27) (cit. on p. 27).

[5] AWS open source. *Intertask Communications*. 2026. URL: https://www.freertos.org/message_passing_performance (visited on 2026-01-27) (cit. on p. 27).

[6] P. Barman, R. Chowdhury, and B. Saha. "Multi-channel secure communication framework for wireless IoT (MCSC-WoT): enhancing security in Internet of Things". In: *Cluster Computing* 28.11 (2025), p. 691 (cit. on pp. 14–16, 18, 19).

[7] C. Bormann, A. P. Castellani, and Z. Shelby. "CoAP: An Application Protocol for Billions of Tiny Internet Nodes". In: *IEEE Internet Computing* 16.2 (2012), pp. 62–67. DOI: 10.1109/MIC.2012.29 (cit. on p. 7).

[8] R. Casadei et al. "Engineering Resilient Collaborative Edge-Enabled IoT". In: *2019 IEEE International Conference on Services Computing (SCC)*. 2019, pp. 36–45. DOI: 10.1109/SCC.2019.00019 (cit. on pp. 12, 13).

[9] M. Chui, M. Collins, and M. Patel. *The Internet of Things: Catching up to an accelerating opportunity*. McKinsey & Company. 2021-11. URL: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/iot-value-set-to-accelerate-through-2030-where-and-how-to-capture-it (visited on 2025-12-26) (cit. on p. 1).

[10] S. Das et al. "Securing IoT-Based Smart Healthcare Systems by Using Advanced Lightweight Privacy-Preserving Authentication Scheme". In: *IEEE Internet of Things Journal* 10.21 (2023), pp. 18486–18494. DOI: 10.1109/JIOT.2023.3283347 (cit. on p. 19).

[11] A. Dauda, O. Flauzac, and F. Nolot. "A Survey on IoT Application Architectures". In: *Sensors* 24.16 (2024). ISSN: 1424-8220. DOI: 10.3390/s24165320 (cit. on p. 1).

[12] B. Dorsemaine et al. "Internet of Things: A Definition & Taxonomy". In: *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*. 2015, pp. 72–77. DOI: 10.1109/NGMAST.2015.71 (cit. on p. 5).

[13] Espressif Systems. *ESP IoT Development Framework*. 2026. URL: https://www.espressif.com/en/products/sdks/esp-idf (visited on 2026-01-27) (cit. on p. 27).

[14] S. FakhrHosseini et al. "A taxonomy of home automation: expert perspectives on the future of smarter homes". In: *Information Systems Frontiers* 27.2 (2025), pp. 449–466. DOI: 10.1007/s10796-024-10496-9 (cit. on p. 5).

[15] X. Fan and G. Gong. "LPKM: A Lightweight Polynomial-Based Key Management Protocol for Distributed Wireless Sensor Networks". In: *Ad Hoc Networks*. Ed. by J. Zheng et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 180–195. ISBN: 978-3-642-36958-2. DOI: 10.1007/978-3-642-36958-2_13 (cit. on p. 19).

[16] P. Fouto et al. "Babel: A Framework for Developing Performant and Dependable Distributed Protocols". In: *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. 2022, pp. 146–155. DOI: 10.1109/SRDS55811.2022.00022 (cit. on pp. 3, 26).

[17] C. Geeng and F. Roesner. "Who's In Control? Interactions In Multi-User Smart Homes". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–13. ISBN: 9781450359702. DOI: 10.1145/3290605.3300498 (cit. on p. 5).

[18] P. Gope and B. Sikdar. "Lightweight and Privacy-Preserving Two-Factor Authentication Scheme for IoT Devices". In: *IEEE Internet of Things Journal* 6.1 (2019), pp. 580–589. DOI: 10.1109/JIOT.2018.2846299 (cit. on p. 19).

[19] S. Hamdan, M. Ayyash, and S. Almajali. "Edge-Computing Architectures for Internet of Things Applications: A Survey". In: *Sensors* 20.22 (2020). ISSN: 1424-8220. DOI: 10.3390/s20226441 (cit. on p. 1).

[20] J. Han et al. "Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption". In: *IEEE Transactions on Parallel and Distributed Systems* 23.11 (2012), pp. 2150–2162. DOI: 10.1109/TPDS.2012.50 (cit. on p. 19).

[21] K. Khanchuea and R. Siripokarpirom. "A Multi-Protocol IoT Gateway and WiFi/BLE Sensor Nodes for Smart Home and Building Automation: Design and Implementation". In: *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*. 2019, pp. 1–6. DOI: 10.1109/ICTEmSys.2019.8695968 (cit. on pp. 9, 12, 14, 15).

[22] D. Kreković et al. "Reducing communication overhead in the IoT–edge–cloud continuum: A survey on protocols and data reduction strategies". In: *Internet of Things* 31 (2025), p. 101553. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2025.101553 (cit. on p. 1).

[23] J. Leitao, J. Pereira, and L. Rodrigues. "HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast". In: *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. 2007, pp. 419–429. DOI: 10.1109/DSN.2007.56 (cit. on pp. 10, 12, 17, 18, 30).

[24] H. Li, K. Ota, and M. Dong. "Always Connected Things: Building Disaster Resilience IoT Communications". In: *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. 2019, pp. 570–577. DOI: 10.1109/ICPADS47876.2019.00087 (cit. on pp. 13, 15).

[25] C.-C. Lin, S. Shieh, and J.-C. Lin. "Lightweight, Distributed Key Agreement Protocol for Wireless Sensor Networks". In: *2008 Second International Conference on Secure System Integration and Reliability Improvement*. 2008, pp. 96–102. DOI: 10.1109/SSIRI.2008.30 (cit. on p. 19).

[26] M5Stack. *ESP32 Core Basic*. 2026. URL: https://docs.m5stack.com/en/core/basic (visited on 2026-01-30) (cit. on p. 32).

[27] P. Maciel et al. "A survey on reliability and availability modeling of edge, fog, and cloud computing". In: *Journal of Reliable Intelligent Environments* 8.3 (2022), pp. 227–245 (cit. on pp. 1, 2).

[28] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh. "Reliability and high availability in cloud computing environments: a reference roadmap". In: *Human-centric Computing and Information Sciences* 8.1 (2018), p. 20 (cit. on p. 1).

[29] R. Montella, M. Ruggieri, and S. Kosta. "A fast, secure, reliable, and resilient data transfer framework for pervasive IoT applications". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2018, pp. 710–715. DOI: 10.1109/INFCOMW.2018.8406884 (cit. on pp. 11, 13).

[30] J. Moy. *OSPF Version 2*. RFC 1247. 1991-07. DOI: 10.17487/RFC1247 (cit. on p. 23).

[31] M. Nakkar, R. AlTawy, and A. Youssef. "GASE: A Lightweight Group Authentication Scheme With Key Agreement for Edge Computing Applications". In: *IEEE Internet of Things Journal* 10.1 (2023), pp. 840–854. DOI: 10.1109/JIOT.2022.3204335 (cit. on p. 19).

[32] R. Narayanan and C. S. R. Murthy. "A Routing Framework With Protocol Conversions Across Multiradio IoT Platforms". In: *IEEE Internet of Things Journal* 8.6 (2021), pp. 4417–4432. DOI: 10.1109/JIOT.2020.3028239 (cit. on p. 15).

[33] J. Pan and J. McElhannon. "Future Edge Cloud and Edge Computing for Internet of Things Applications". In: *IEEE Internet of Things Journal* 5.1 (2018), pp. 439–449. DOI: 10.1109/JIOT.2017.2767608 (cit. on p. 2).

[34] J. Porte et al. "Heterogeneous wireless IoT architecture for natural disaster monitorization". In: *EURASIP Journal on Wireless Communications and Networking* 2020.1 (2020), p. 184 (cit. on pp. 10, 12, 14, 15).

[35] Raspberry Pi Ltd. *Raspberry Pi Pico SDK*. 2026. URL: https://www.raspberrypi.com/documentation/pico-sdk/index_doxygen.html (visited on 2026-01-27) (cit. on p. 27).

[36] J. Ren et al. "Collaborative Cloud and Edge Computing for Latency Minimization". In: *IEEE Transactions on Vehicular Technology* 68.5 (2019), pp. 5031–5044. DOI: 10.1109/TVT.2019.2904244 (cit. on pp. 1, 2).

[37] K. Scott and S. C. Burleigh. *Bundle Protocol Specification*. RFC 5050. 2007-11. DOI: 10.17487/RFC5050 (cit. on p. 11).

[38] J. P. Shanmuga Sundaram et al. "MARS: Multi-radio Architecture with ML-powered Radio Selection for Mesoscale IoT Applications". In: *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2025, pp. 1–8. DOI: 10.1109/ETFA65518.2025.11205544 (cit. on p. 15).

[39] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. 2014-06. DOI: 10.17487/RFC7252 (cit. on p. 7).

[40] S. Sinha. "State of IoT 2025: Number of connected IoT devices growing 14% to 21.1 billion globally". In: *IoT Analytics* (2025) (cit. on p. 1).

[41] A. Staff. *Summary of the Amazon DynamoDB Service Disruption in the Northern Virginia (US-EAST-1) Region*. 2025. URL: https://aws.amazon.com/message/101925/ (visited on 2025-12-26) (cit. on p. 1).

[42] I. R. Team. *AWS Outage Analysis: October 20, 2025*. 2025. URL: https://www.thousandeyes.com/blog/aws-outage-analysis-october-20-2025 (visited on 2025-12-26) (cit. on pp. 1, 21).

[43] T. Theodorou et al. "A Multi-Protocol Software-Defined Networking Solution for the Internet of Things". In: *IEEE Communications Magazine* 57.10 (2019), pp. 42–48. DOI: 10.1109/MCOM.001.1900056 (cit. on p. 15).

[44] H. T. T. Truong et al. "Towards Secure and Decentralized Sharing of IoT Data". In: *2019 IEEE International Conference on Blockchain (Blockchain)*. 2019, pp. 176–183. DOI: 10.1109/Blockchain.2019.00031 (cit. on p. 19).

[45] S. Voulgaris, D. Gavidia, and M. Van Steen. "Cyclon: Inexpensive membership management for unstructured p2p overlays". In: *Journal of Network and systems Management* 13.2 (2005), pp. 197–217 (cit. on p. 10).

[46] J. Won et al. "Decentralized Public Key Infrastructure for Internet-of-Things". In: *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. 2018, pp. 907–913. DOI: 10.1109/MILCOM.2018.8599710 (cit. on p. 19).

[47] L. Xing. "Reliability in Internet of Things: Current Status and Future Perspectives". In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 6704–6721. DOI: 10.1109/JIOT.2020.2993216 (cit. on p. 2).

[48] N. Xiong et al. "Randomized and Efficient Time Synchronization in Dynamic Wireless Sensor Networks: A Gossip-Consensus-Based Approach". In: *Complexity* 2018.1 (2018), p. 4283087. DOI: 10.1155/2018/4283087 (cit. on pp. 17, 18, 30).

[49] H. Yıldız, M. Cenk, and E. Onur. "PLGAKD: A PUF-Based Lightweight Group Authentication and Key Distribution Protocol". In: *IEEE Internet of Things Journal* 8.7 (2021), pp. 5682–5696. DOI: 10.1109/JIOT.2020.3032757 (cit. on p. 19).

[50] W. Yu et al. "A Survey on the Edge Computing for the Internet of Things". In: *IEEE Access* 6 (2018), pp. 6900–6919. DOI: 10.1109/ACCESS.2017.2778504 (cit. on p. 1).