



**JOÃO RICARDO BOGALHO BRILHA**  
BSc in Computer Science and Engineering

## MICROBABEL

A MULTI-PROTOCOL APPROACH FOR RESILIENT  
AND DECENTRALIZED IOT NETWORKS

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

NOVA University Lisbon  
*Draft: January 29, 2026*



## MICROBABEL

A MULTI-PROTOCOL APPROACH 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: January 29, 2026*

## ABSTRACT

Regardless of the language in which the dissertation is written, usually there are at least two abstracts: one abstract in the same language as the main text, and another abstract in some other language.

The abstracts' order varies with the school. If your school has specific regulations concerning the abstracts' order, the NOVAthesis L<sup>A</sup>T<sub>E</sub>X (`novathesis`) (L<sup>A</sup>T<sub>E</sub>X) template will respect them. Otherwise, the default rule in the `novathesis` template is to have in first place the abstract in *the same language as main text*, and then the abstract in *the other language*. For example, if the dissertation is written in Portuguese, the abstracts' order will be first Portuguese and then English, followed by the main text in Portuguese. If the dissertation is written in English, the abstracts' order will be first English and then Portuguese, followed by the main text in English. However, this order can be customized by adding one of the following to the file `5_packages.tex`.

```
\ntsetup{abstractorder={<LANG_1>, ..., <LANG_N>}}  
\ntsetup{abstractorder={<MAIN_LANG>={<LANG_1>, ..., <LANG_N>}}}
```

For example, for a main document written in German with abstracts written in German, English and Italian (by this order) use:

```
\ntsetup{abstractorder={de={de,en,it}}}
```

Concerning its contents, the abstracts should not exceed one page and may answer the following questions (it is essential to adapt to the usual practices of your scientific area):

1. What is the problem?
2. Why is this problem interesting/challenging?
3. What is the proposed approach/solution/contribution?
4. What results (implications/consequences) from the solution?

**Keywords:** One keyword · Another keyword · Yet another keyword · One keyword more · The last keyword

# RESUMO

Independentemente da língua em que a dissertação está escrita, geralmente esta contém pelo menos dois resumos: um resumo na mesma língua do texto principal e outro resumo numa outra língua.

A ordem dos resumos varia de acordo com a escola. Se a sua escola tiver regulamentos específicos sobre a ordem dos resumos, o template (L<sup>A</sup>T<sub>E</sub>X) **novathesis** irá respeitá-los. Caso contrário, a regra padrão no template **novathesis** é ter em primeiro lugar o resumo *no mesmo idioma do texto principal* e depois o resumo *no outro idioma*. Por exemplo, se a dissertação for escrita em português, a ordem dos resumos será primeiro o português e depois o inglês, seguido do texto principal em português. Se a dissertação for escrita em inglês, a ordem dos resumos será primeiro em inglês e depois em português, seguida do texto principal em inglês. No entanto, esse pedido pode ser personalizado adicionando um dos seguintes ao arquivo `5_packages.tex`.

```
\abstractorder(<MAIN_LANG>) := {<LANG_1>, ..., <LANG_N>}
```

Por exemplo, para um documento escrito em Alemão com resumos em Alemão, Inglês e Italiano (por esta ordem), pode usar-se:

```
\ntsetup{abstractorder={de={de,en,it}}}
```

Relativamente ao seu conteúdo, os resumos não devem ultrapassar uma página e frequentemente tentam responder às seguintes questões (é imprescindível a adaptação às práticas habituais da sua área científica):

1. Qual é o problema?
2. Porque é que é um problema interessante/desafiante?
3. Qual é a proposta de abordagem/solução?
4. Quais são as consequências/resultados da solução proposta?

**Palavras-chave:** Primeira palavra-chave · Outra palavra-chave · Mais uma palavra-chave · A última palavra-chave

# CONTENTS

<b>Listings</b>	vii
<b>Glossary</b>	viii
<b>Acronyms</b>	ix
<b>1 Introduction</b>	1
<b>2 Related Work</b>	6
2.1 Wireless Communication Technologies . . . . .	8
2.1.1 Application-Layer Protocols . . . . .	8
2.1.2 Physical- and Link-Layer Technologies . . . . .	8
2.1.3 Discussion . . . . .	9
2.2 Peer-to-peer (P2P) Mesh Networking and Topology Management . . . . .	10
2.2.1 Peer Discovery and Topology Maintenance . . . . .	10
2.2.2 Routing in Partitioned and Intermittently Connected Networks . .	12
2.2.3 Limitations of Centralized Coordination . . . . .	12
2.2.4 Discussion . . . . .	13
2.3 Multi-Protocol Communication and Adaptive Protocol Selection . . . . .	14
2.3.1 Multi-Channel Resilience Architectures . . . . .	14
2.3.2 Protocol Heterogeneity on Embedded Platforms . . . . .	15
2.3.3 Runtime Adaptive Protocol Selection . . . . .	15
2.3.4 Discussion . . . . .	16
2.4 Decentralized Synchronization and Time Coordination . . . . .	16
2.4.1 Master-Based Synchronization as Counterexample . . . . .	17
2.4.2 Gossip-Based Masterless Synchronization . . . . .	17
2.4.3 Coordination Through Passive View Maintenance . . . . .	18
2.4.4 Discussion . . . . .	18
2.5 Lightweight Data Compression and Energy Efficiency . . . . .	19
2.5.1 Prediction-Based Data Reduction . . . . .	19

2.5.2	Lightweight Compression . . . . .	20
2.5.3	Receiver-Side Energy Considerations . . . . .	22
2.5.4	Discussion . . . . .	22
2.6	Security Considerations and Scope Delimitation . . . . .	23
2.7	Summary . . . . .	24
<b>3</b>	<b>Solution Architecture</b>	<b>26</b>
3.1	Recap . . . . .	26
3.1.1	The Problem . . . . .	26
3.1.2	Technical Challenges . . . . .	26
3.2	System Requirements . . . . .	27
3.2.1	Overview of $\mu$ -Babel’s Approach . . . . .	27
3.3	System Assumptions and Threat Model . . . . .	28
3.3.1	Operational Assumptions . . . . .	29
3.3.2	Threat Model . . . . .	29
3.3.3	Explicitly Out of Scope . . . . .	29
3.4	Hardware Platform . . . . .	30
3.4.1	Class-1: Sensor Nodes . . . . .	30
3.4.2	Class-2: Aggregation Nodes . . . . .	32
3.5	Software Stack and Development Environment . . . . .	32
3.5.1	Operating System Layer . . . . .	33
3.5.2	Platform SDKs and Hardware Abstractions . . . . .	33
3.5.3	Implementation Language . . . . .	33
3.5.4	Concurrency Model . . . . .	33
3.6	System Model . . . . .	34
3.6.1	Architecture . . . . .	34
<b>4</b>	<b>Progress Report and Planning</b>	<b>39</b>
4.1	Progress Report . . . . .	39
4.2	Planning . . . . .	39
4.2.1	Calendarization – GANTT . . . . .	39
<b>Bibliography</b>		<b>40</b>

## LIST OF FIGURES

## LIST OF TABLES

2.1	Protocol Characteristics . . . . .	9
2.2	Related work summary . . . . .	24
3.1	Class-1 Device Specifications . . . . .	30
3.2	Supported Communication Protocols . . . . .	31
3.3	Class-2 Device Specifications . . . . .	32

## **L**I**S**TIN*G*S

## GLOSSARY

<b>ESP-NOW</b>	Wireless communication protocol developed by Espressif. Enables direct and low-power communication between ESP-family devices without the need for a router. ( <i>pp. 3, 4, 12, 31, 32, 36</i> )
<b>hysteresis</b>	Use of "decision inertia" or asymmetric thresholds to prevent frequent switching in response to small or short-lived changes in system metrics ( <i>pp. 12, 36, 37</i> )
<b>LoRa</b>	From "long range", proprietary radio communication technology ( <i>pp. 3, 4, 10, 16, 31, 32, 35–38</i> )
<b>Modbus</b>	An application-layer client/server serial communication protocol, widely used in industrial automation systems. ( <i>p. 12</i> )
<b>MQTT</b>	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 ( <i>pp. 12, 13, 19, 28, 33</i> )
<b>RPi</b>	Raspberry Pi full-size boards ( <i>p. 21</i> )
<b>Thread</b>	Low-power, mesh networking technology based on the IEEE 802.15.4 standard. Increasingly adopted for smart-home applications. ( <i>p. 31</i> )
<b><math>\mu</math>-Babel</b>	MicroBabel ( <i>pp. iv, 2–4, 6, 10, 13, 16, 19, 20, 22–25, 27, 28, 30, 32–39</i> )
<b>ZigBee</b>	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. ( <i>pp. 3, 4, 12, 31, 32</i> )

## ACRONYMS

<b>ABE</b>	Attribute-based encryption ( <i>p.</i> 23)
<b>AES</b>	Advanced Encryption Standard ( <i>pp.</i> 11, 14, 23, 24, 29, 40)
<b>API</b>	Application programming interface ( <i>p.</i> 34)
<b>ARIMA</b>	Autoregressive integrated moving average ( <i>pp.</i> 17, 19)
<b>AS</b>	Authentication Server ( <i>pp.</i> 21, 24)
<b>BLE</b>	Bluetooth Low Energy ( <i>pp.</i> 3, 4, 7, 10, 12, 16, 31, 32, 34–38)
<b>CA</b>	Certificate authority ( <i>pp.</i> 6, 20, 23)
<b>CoAP</b>	Constrained Application Protocol ( <i>pp.</i> 12, 13, 19, 28, 33)
<b>CPU</b>	Central processing unit ( <i>pp.</i> 30, 33)
<b>CR</b>	Compression ratio ( <i>p.</i> 18)
<b>DNS</b>	Domain Name System ( <i>p.</i> 1)
<b>DODAG</b>	Destination-oriented directed acyclic graph ( <i>pp.</i> 8, 35)
<b>DTN</b>	Delay-tolerant networking ( <i>pp.</i> 8, 11)
<b>ESP-IDF</b>	Espressif IoT Development Framework ( <i>p.</i> 34)
<b>EWMA</b>	Exponentially weighted moving average ( <i>p.</i> 12)
<b>FHSS</b>	Frequency-hopping spread spectrum ( <i>pp.</i> 11, 14)
<b>FIRE</b>	Fast Integer REgression ( <i>p.</i> 18)
<b>GASE</b>	Group Authentication Scheme at the Edge ( <i>pp.</i> 20, 24, 39)
<b>GL</b>	Group leader ( <i>pp.</i> 21, 24, 30, 39)
<b>GPIO</b>	General-purpose input/output ( <i>p.</i> 34)
<b>HAL</b>	Hardware Abstraction Layer ( <i>p.</i> 34)
<b>HF</b>	High frequency ( <i>p.</i> 11)

<b>HTTP</b>	Hypertext Transfer Protocol ( <i>pp. 9, 12, 13, 19</i> )
<b>HVAC</b>	Heating, ventilation, and air conditioning ( <i>p. 5</i> )
<b>I<sup>2</sup>C</b>	Inter-Integrated Circuit ( <i>p. 34</i> )
<b>IoT</b>	Internet of Things ( <i>pp. 1–3, 5–11, 16, 19–21, 23, 26, 28, 30, 31, 36</i> )
<b>KDC</b>	Key distribution center ( <i>pp. 6, 20, 22, 24, 39</i> )
<b>LE</b>	Low Energy ( <i>p. 31</i> )
<b>LoRaWAN</b>	LoRa Wide Area Network ( <i>pp. 7, 11, 13, 36</i> )
<b>LPKM</b>	Lightweight Polynomial-based Key Management ( <i>pp. 22, 24, 40</i> )
<b>LPWAN</b>	Low-Power Wide Area Network ( <i>p. 11</i> )
<b>MAC</b>	Message Authentication Code ( <i>pp. 20, 21, 29, 30, 39</i> )
<b>MDL</b>	Minimum description length ( <i>p. 19</i> )
<b>novathesis</b>	NOVAthesis L <sup>A</sup> T <sub>E</sub> X ( <i>pp. i, ii</i> )
<b>NTP</b>	Network Time Protocol ( <i>p. 15</i> )
<b>NVIS</b>	Near Vertical Incidence Skywave ( <i>pp. 11, 13, 36</i> )
<b>OS</b>	Operating system ( <i>pp. 33, 34</i> )
<b>OSPF</b>	Open Shortest Path First ( <i>p. 28</i> )
<b>P2P</b>	Peer-to-peer ( <i>pp. iii, 6, 7, 9, 10, 12, 27, 30, 31, 35, 36, 38</i> )
<b>PDA</b>	Personal digital assistant ( <i>p. 22</i> )
<b>PKI</b>	Public key infrastructure ( <i>pp. 23, 24</i> )
<b>PLR</b>	Packet loss ratio ( <i>pp. 8, 36</i> )
<b>PRNG</b>	Pseudorandom number generator ( <i>pp. 11, 14</i> )
<b>PSRAM</b>	Pseudostatic random-access memory ( <i>p. 30</i> )
<b>PUF</b>	Physically Unclonable Function ( <i>pp. 21, 24, 27</i> )
<b>QoS</b>	Quality of Service ( <i>pp. 4, 6, 12</i> )
<b>RAM</b>	Random-access memory ( <i>p. 33</i> )
<b>RF</b>	Radio frequency ( <i>pp. 10, 29</i> )
<b>RGCS</b>	Randomized gossip-consensus-based sync ( <i>pp. 14–16, 25, 37, 38, 40</i> )
<b>RLE</b>	Run-length encoding ( <i>pp. 18, 19, 38, 39</i> )
<b>RPL</b>	Routing Protocol for Low-Power and Lossy Networks ( <i>pp. 10, 11, 35</i> )
<b>RSSI</b>	Received signal strength indicator ( <i>pp. 7, 12, 36</i> )
<b>RTOS</b>	Real-time operating system ( <i>p. 33</i> )

<b>SDK</b>	Software Development Kit ( <i>pp. 33, 34</i> )
<b>SDN</b>	Software-defined networking ( <i>pp. 6, 13, 24, 36</i> )
<b>SF</b>	Spreading Factor ( <i>p. 32</i> )
<b>SPI</b>	Serial Peripheral Interface ( <i>pp. 31, 34</i> )
<b>SRAM</b>	Static random-access memory ( <i>p. 30</i> )
<b>SSID</b>	Service Set Identifier ( <i>p. 32</i> )
<b>SSS</b>	Shamir Secret Sharing ( <i>p. 20</i> )
<b>TCP</b>	Transmission Control Protocol ( <i>pp. 6, 8, 25, 35</i> )
<b>UART</b>	Universal asynchronous receiver-transmitter ( <i>p. 34</i> )
<b>WSN</b>	Wireless Sensor Network ( <i>pp. 8, 11, 14</i> )

## 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 [41], with industry analysts estimating that IoT technologies could unlock between \$5.5 and \$12.6 trillion in economic value by the same year [10].

However, current IoT systems remain fundamentally dependent on continuous Internet connectivity and centralized cloud infrastructures [52, 21]. 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 [44] 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 [42].

Attempts have been made to address these challenges through incremental improvements, such as shifting focus to edge computing in order to reduce latency [21, 52, 37], redundant cloud regions for availability [29, 28], and hybrid architectures that combine local and remote processing [1, 12, 24].

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  **$\mu$ -Babel** addresses such scenarios across scales: 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 [37, 33], reduced availability during network disruptions or cloud outages [49, 28], 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.

The Babel Ecosystem [18] 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  **$\mu$ -Babel**, a lightweight framework targeting embedded platforms (e.g., ESP32, Raspberry Pi Pico) aimed at developing resilient, multi-protocol, and decentralized IoT systems that can operate autonomously during infrastructure failures.

**$\mu$ -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"

**Note on terminology:** Throughout this work the term *multi-protocol* is used to refer to heterogeneity in communication technologies (i.e., Bluetooth Low Energy (BLE), LoRa, Wi-Fi, etc.) that have distinct link/physical layers, while *multi-channel* refers to frequency diversity within a single protocol (e.g., 2.4GHz Wi-Fi channels 1-13, LoRa frequency hopping). Multi-protocol operation provides technology diversity for resilience; multi-channel operation provides frequency diversity for both anti-jamming and additional throughput.

## 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.  **$\mu$ -Babel** will address this by leveraging a multi-protocol communication approach, supporting a diverse set of wireless protocols that can operate independently of infrastructure.

By enabling adaptive protocol 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,  **$\mu$ -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.  **$\mu$ -Babel** will provide automatic peer discovery across multiple protocols, 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.

## Expected Contributions

We plan to make the following main contributions:

- A decentralized architecture supporting multiple communication protocols (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-protocol 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

The remainder of this document is arranged as follows:

**Chapter 2** Reviews widely-used communication protocols and related work in P2P networking, multi-protocol communication, decentralized synchronization, data compression, and 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.

merge with  
chap3 se não  
conseguir re-  
duzir espaço  
elsewhere....

## 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:

**IoT systems** [14] 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** [19, 16] 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.

**em vez de "iot systems": smart systems?** The challenges faced by disaster-resilient IoT systems 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, or 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.

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 five 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 master nodes; (Section 2.5) lightweight data compression and energy optimization techniques to extend autonomous operation duration and device lifetime.

For each area, we examine how existing approaches handle (or fail to handle) infrastructure failure scenarios, identify architectural assumptions that conflict with disaster-resilient requirements, and position  $\mu$ -Babel’s contributions relative to the current state of the art.

Additionally, security and privacy considerations are presented in Section 2.6, which

FIXED: updated security section description

outlines the scope of this work in regard to higher-layer safety mechanisms.

## 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.

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

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) [8, 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  **$\mu$ -Babel**. For this reason, while such protocols are of interest for broader interoperability (see Non-Functional Requirements in Section 3.2.1), they are not a part of the foundational protocols employed by this 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. While the previous section addressed higher-layer messaging protocols, the technologies discussed here provide the foundational communication stack that determines the behavior of  **$\mu$ -Babel**.

Table 2.1 details the basic operational characteristics of the wireless communication technologies considered in this work.

get actual  
power draw  
values from  
somewhere

Table 2.1: Protocol Characteristics

Protocol	Discovery	Range <sup>1</sup>	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	Low
ESP-NOW	Peer list	50-100 m	2.4 GHz	1 Mbps	Very Low
ZigBee	Beacon	10-100 m	2.4 GHz	250 Kbps	Very Low
LoRa	Preamble detection	2-10 km	433/868/915 MHz <sup>2</sup>	0.3-50 Kbps <sup>3</sup>	Low <sup>3</sup>

<sup>1</sup> Outdoor line-of-sight; indoor ranges typically 30-50% lower

<sup>2</sup> Frequency depends on regional regulations

<sup>3</sup> Data rate and power consumption depend on Spreading Factor (SF); higher SF = longer range, lower rate, higher energy cost

These characteristics drive protocol selection based on deployment context and communication requirements:

**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.  $\mu$ -Babel's hybrid discovery mechanism (Section 3.6.1) will accommodate these differences while maintaining a unified peer view across protocols.

**Energy Considerations:** Protocol selection must account for energy costs: Wi-Fi consumes significantly more power than BLE, while LoRa power consumption depends heavily on SF configuration. The data compression mechanisms (Section 3.6.1) interact with protocol selection to minimize overall energy expenditure.

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 communication technologies highlight that no single protocol can simultaneously optimize for range, throughput, energy efficiency, and autonomy under infrastructure failures. Instead, resilience emerges from the ability to combine the complimentary

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-protocol operation and architectural choices discussed in Chapter 3.

## 2.2 P2P Mesh Networking and Topology Management

Infrastructure-dependent star topologies in which devices communicate via a central coordinator or gateway suffer from a similar downside when compared with single-protocol 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.

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

As discussed in Section 2.3.2, the Multi-Protocol IoT Gateway implementation [23] demonstrates BLE-based neighbor discovery with Received signal strength indicator (RSSI) measurements for proximity estimation.

While that work focuses on multi-protocol integration, 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.

This restriction limits route diversity and resilience, creating dependency chains where a single intermediate node failure can disconnect entire subtrees.

A more sophisticated approach to membership management is presented in HyParView [25], in which each node maintains two distinct partial views for scalability: a small active view ( $\text{size} = \text{fanout} + 1$ ) containing nodes with which symmetric links are

FIXED: justify  
this claim

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 [47].

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.

FIXED: adicionei isto com base na nota. mas nós vamos ter some form of multi-hop também so not sure I understand pqq é mau

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.

### 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 [38] 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 [30] 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 [9] addresses coordinator failures through dynamic leader election and backup mechanisms within their framework.

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

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

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.

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.

queremos min-gle com o Ba-be... se calhar não entrar por aqui ou tudo bem porque não vamos usar os big-raspis para coordination?

#### 2.2.4 Discussion

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

HyParView's [25] 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 [23] 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 [30] enables partition tolerance but targets cloud-backed deployments rather than P2P mesh scenarios.

Coordination frameworks like the one in [9] 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.

$\mu$ -Babel's approach (Section 3.6.1) must adapt HyParView's architectural principles to BLE/LoRa connectionless communication while supporting multi-hop routing without depending on coordinators or routing infrastructure.

## 2.3 Multi-Protocol Communication and Adaptive Protocol Selection

Single-protocol (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 damage to access points which may become unavailable as a result.

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

### 2.3.1 Multi-Channel Resilience Architectures

Several disaster-focused systems explicitly address infrastructure failures through the exploration of protocol diversity.

The AWCT (Always Connected Things) framework [26] 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), 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 [23] shows the technical feasibility of multi-protocol 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.

Their implementation uses BLE advertising beacons for neighbor discovery with 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. The system also supports automatic parent reselection when the current parent becomes unreachable, making it somewhat resilient to single node failures.

Testing demonstrated successful multi-hop operation up to 5 hops in office environments, but evaluation was limited to linear network topologies. Additionally, this architecture relies on a tree topology with master-slave communication that is centered around a gateway (coordinator) rather than being a P2P mesh, creating a single point of failure at the coordinator node.

### 2.3.3 Runtime Adaptive Protocol Selection

The MINOS platform [45] 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

FIXED: removed MDPI paper

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.

#### 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 [26] 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 [23] uses tree topologies with gateway coordinators as single points of failure, and MINOS [45] assumes SDN controllers with persistent MQTT/web infrastructure.

We have found no work that addresses fully distributed multi-protocol coordination where nodes can autonomously select and switch protocols without resorting to persistent infrastructure.  $\mu$ -Babel’s approach to decentralized protocol selection is detailed in Section 3.6.1 along with masterless synchronization mechanisms.

## 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 to overcome 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

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 [50] addresses time synchronization in dynamic WSNs through randomized asynchronous gossip. Each node maintains a logical clock ( $T$ ) composed of rate ( $\alpha$ ) and offset ( $\beta$ ) parameters, which together transform the node’s hardware clock ( $\tau$ ) 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  $\lambda$ .

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  $\beta$  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 [25] 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.

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

Master-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 [50] 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 [25] shuffle-based passive view maintenance provides complementary coordination through periodic peer exchanges that could carry piggybacked sync messages, but was not designed for time synchronization integration. One example would

FIXED: moved  
NTP and GPS  
talk to discussion

FIXED: aqui

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), Global Positioning System (GPS) clock synchronization, or simply those maintaining more reliable clock sources.

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.

$\mu$ -Babel's approach (Section 3.6.1) will strive to extend RGCS to multi-protocol scenarios with per-protocol Poisson processes while preventing protocol selection oscillations during sync events.

## 2.5 Lightweight Data Compression and Energy Efficiency

Energy constraints present one of the most fundamental limitations in battery-powered IoT deployments, particularly in hazardous deployments where replacements and recharging are impractical, and during disaster scenarios where grid power likely becomes unavailable.

While multi-protocol communication and decentralized coordination efforts provide resilience, they also introduce increased energy costs through frequent transmissions, protocol adaptation overhead and forwarding costs.

Data transmission dominates energy consumption across wireless technologies: LoRa and SIGFOX can represent up to 99.9% of power consumption for transmission, while even shorter-range technologies like BLE and IEEE 802.15.4 dedicate 85-90% of their energy budget to radio operations [43].

This reality makes data reduction techniques essential for extending operational lifetime in infrastructure-independent scenarios. However, compression itself consumes energy through additional computational work, thus creating an unavoidable tug-of-war between compression overhead and transmission benefits.

### 2.5.1 Prediction-Based Data Reduction

The Ambrosia protocol [43] demonstrates that lightweight prediction can achieve substantial data reductions on resource-constrained devices. The core idea is that not all sensor readings need to be transmitted if their values can be accurately predicted at the server (within the bounds of a specified error threshold), provided that both sender and receiver maintain a synchronized "prediction state".

Their approach uses window-based forecasting where the next sample is predicted

by adding the average differences between recent previous samples, making it dramatically lighter than sophisticated time-series forecasting techniques like Autoregressive integrated moving average (ARIMA) [40] while still achieving comparable data reduction performance.

The protocol works by sending the first  $w$  (window size) true samples collected by the sensor node to the server, and for every sample after that comparing its true value to the one predicted locally; the true sample value is sent to the server only if the absolute difference between it and the prediction is greater than a user or application-specified error threshold  $\delta$ .

The same simple prediction scheme is used on both ends of this communication, and the predicted value is the one used for further predictions – even on the sensor node that knows the true value, to ensure consistency between the two endpoints.

This design achieves up to 60% data reduction with appropriate  $\delta$  configuration while maintaining sufficient accuracy for diverse applications: for error-tolerant use cases like anomaly detection,  $\delta$  values will be higher to allow for reasonable fluctuations, thus enabling more significant data reductions, while error-sensitive applications requiring higher precision will naturally lean towards stricter error thresholds, but still benefitting from some amount of data reduction.

The major takeaway from this work is that the window-based prediction executed 99% faster than ARIMA forecasting in their evaluation, making it viable for streaming sensor data at high rates. The approach achieved 2 $\times$  battery lifetime improvement in high-traffic scenarios and demonstrated compatibility with extremely constrained devices (livestock ear tags with Atmel 8-bit microcontrollers).

### 2.5.2 Lightweight Compression

While prediction-based reduction minimizes the number of transmissions, compression techniques are essential to reduce the size of the data that *must* be transmitted.

A few complementary approaches address different considerations and constraints: hybrid schemes balance accuracy with transmission costs through lossy/lossless models; lossless time-series compression exploits the natural temporal relation in continuous sensor readings; and two-tier architectures apply different techniques at both the sensor and gateway level to reduce energy costs.

#### Hybrid Lossy/Lossless Compression

The hybrid compression scheme presented in [13] addresses accuracy requirements by separating real-time lossy transmission from on-demand lossless reconstruction. The Fan algorithm adaptively sub-samples data maintaining bounded error  $\epsilon$ , achieving average 7.8 $\times$  and 2.1 $\times$  Compression ratio (CR) for lossy and lossless compression, respectively.

The approach provides graceful degradation through battery-aware switching, in that when battery drops below a certain threshold (e.g., 20%), the system switches to lossy-only transmission instead of lossless, extending lifetime at the cost of reconstruction accuracy.

This trade-off aligns with disaster-scenario requirements where sustained operation during prolonged periods of network outages often takes higher priority over perfect data-fidelity, as approximate sensor readings over extended periods of time provide greater situational awareness than high-precision measurements from devices that quickly deplete their batteries. However, safety-critical sensors (e.g., structural integrity sensors, hazardous gas detectors) may require different thresholds (if any) or other policies to balance longevity with accuracy requirements.

FIXED: ""é bom? é mau? é adequado?""

### Lossless Time-Series Compression

Sprintz [7] targets lossless compression for multivariate integer time-series with extreme resource constraints, achieving 2-10× compression ratios with <1KB memory footprint and 8-sample block sizes. The four-component algorithm combines forecasting (delta coding or Fast Integer REgression (FIRE) online learning), bit-packing with zigzag encoding, Run-length encoding (RLE) for all-zero blocks, and optional Huffman coding.

Delta coding is extremely fast, and when combined with RLE becomes even more so, as it yields a run of zero errors if the data is constant, which is likely to happen for nominal sensor readings. FIRE forecasting yields better compression, but its online learning approach place it beyond the scope of this paper.

A *forecaster* is employed to predict each sample based on previous ones, and the difference between the next and the predicted sample are encoded. This difference is typically closer to zero than the next sample itself.

Any prediction errors from the previous forecasting step are zigzag encoded as a "payload", and a header is prepended with sufficient information to invert the bit packing.

If a block happens to consist only of zeros, this approach waits for a block in which there is a non-zero error, and the number of all-zero blocks are written out using RLE instead of the (empty) payload.

Finally, the bit packed representation of each block can be *entropy coded* using a Huffman coder, applied to the headers and payloads. This is done after bit packing because not only is it faster, but it also increases compression.

Decompression can achieve up to 3GB/s throughput without Huffman coding (>500MB/s with Huffman) in a single thread, enabling high-speed data retrieval on gateway-tier devices. Compression maintains >200MB/s on 8-bit data, sufficient for real-time operation even on embedded platforms.

### Two-Tier Compression Architecture

The two-tier data reduction technique proposed in [35] applies compression at both sensor nodes (Tier 1) and gateways (Tier 2) to reduce energy consumption in the overall system.

Sensor nodes employ Delta encoding followed by RLE, allowing this approach to achieve 80-84% compression by exploiting the temporal correlation in sensor data. Gateways perform hierarchical clustering based on the Minimum description length (MDL) principle, transmitting *hypothesis* data sets and difference vectors instead of full data sets.

The Delta+RLE implementation – validated through OMNeT++ simulation – proves to be lightweight enough for resource-constrained nodes, while gateway clustering reduces transmission to cloud infrastructure.

### 2.5.3 Receiver-Side Energy Considerations

An often overlooked aspect of IoT energy optimization is the asymmetry between sender and receiver energy consumption. The adaptive protocol selection framework analysis [53] revealed that receiver nodes consistently consume 15-20% more energy than senders across all evaluated protocols (MQTT, CoAP, HTTP).

This asymmetry shifts the energy bottleneck from endpoint sensors to intermediary nodes like gateways and has direct implications for gateway power management in disaster scenarios where battery replacement becomes impossible. The increased receiver cost stems from the added computational work required for parsing, processing, and managing incoming messages, particularly at high message rates.

For  **$\mu$ -Babel**'s proposed heterogeneous architecture where sensor nodes might eventually forward data to aggregators/gateways, this finding has direct implications in that managing the power consumption of these gateways becomes as significant as sensor power optimization, since one gateway might receive substantial data from several nodes even if individually they don't perform frequent transmissions.

### 2.5.4 Discussion

Existing data reduction techniques demonstrate substantial energy savings on resource-constrained platforms but assume single-protocol deployments.

Ambrosia's [43] window-based prediction achieves 60% data reduction with 99% faster execution than ARIMA while maintaining accuracy within application-specific error thresholds, validated on 8-bit microcontrollers.

Sprintz [7] provides lossless time-series compression with 2-10 $\times$  compression ratios using <1KB memory and 8-sample blocks, combining delta coding, zigzag encoding, RLE, and optional Huffman coding with >200MB/s compression throughput suitable for real-time embedded operation.

Hybrid lossy/lossless [13] schemes enable graceful battery-aware degradation, while two-tier architectures [35] (Delta+RLE at sensors, MDL clustering at gateways) achieve 80-84% compression exploiting temporal correlation.

falar mais do  
MDL ou como  
é gateway side  
not so important?

outra vez o pa-  
per do MDPI....  
retirar por com-  
pleto, fingir  
que não vi?

None of these approaches address protocol heterogeneity by themselves, and different wireless technologies have distinct energy costs and range/accuracy trade-offs that warrant protocol-specific compression/reduction strategies.

We have found no work that integrates prediction-based reduction with protocol-aware adaptive thresholding and compression tuning for heterogeneous multi-protocol networks.

$\mu$ -Babel's approach (Section ??) should, therefore, consider lightweight prediction and compression in multi-protocol scenarios with per-protocol thresholds and strategy selection while accounting for gateway receiver energy costs.

TODO: retirar receiver energy stuff se retirar o paper do MDPI

## 2.6 Security Considerations and Scope Delimitation

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  $\mu$ -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) [32] relies on aggregated Message Authentication Codes (MACs) and threshold techniques to reduce computational overhead, while the Lightweight Polynomial-based Key Management (LPKM) protocol [17] 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) [20, 51, 11]. 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 [27] 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 [46] and decentralized Public key infrastructures (PKIs) [48]. 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 [22].

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  **$\mu$ -Babel** framework focuses on providing a foundation for resilient and decentralized communication on top of which security mechanisms may be integrated.

## 2.7 Summary

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

Table 2.2: Related work summary

Domain	Related Work Limitations	$\mu$ -Babel Approach
Protocol coordination	Master nodes (MCSC-WoT), SDN controllers (MINOS), gateway coordinators (multi-protocol gateway)	Fully distributed selection with hysteresis control
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, multi-graph converge-to-max
Compression	Single-protocol optimization (Ambrosia, Sprintz), sender-focused	Protocol-aware adaptive thresholds, gateway receiver energy management

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, and authentication protocols depend on centralized servers or group leaders.

Even distributed approaches like HyParView and RGCS make assumptions (routing infrastructure, persistent TCP connections, single-protocol homogeneity) incompatible

FIXED: removed security row

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.

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

FIXED: closing paragraph, mais narrativa etc?

# 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 [44] 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:** Protocol selection requires master nodes or 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-protocol operation:** Gossip-based synchronization assumes homogeneous networks where all nodes share identical communication characteristics. Compression techniques optimize for single technologies without considering protocol-specific energy costs. Topology maintenance protocols assume either persistent connections or uniform radio properties ill-suited for multi-protocol embedded platforms.

**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.

**$\mu$ -Babel** will address these interconnected challenges through a unified framework designed specifically for resource-constrained multi-protocol 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 $\mu$ -Babel's Approach

In order to effectively overcome the aforementioned challenges, we can highlight a set of functional requirements that  **$\mu$ -Babel** must meet in order to eliminate the *dependency* on centralized coordination, continuous connectivity, and cloud infrastructure; and a set of non-functional requirements which would strengthen the overall quality of the framework and ensure it can be widely adopted with ease.

or (non-)functional requirements

#### Functional Requirements

"something  
something  
these are es-  
sential ..." ou o  
parágrafo ante-  
rior basta?

**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 coordinators and static master nodes. Instead, local P2P gossip interactions will be the basis for coordination and synchronization.

**Multi-Protocol Exploitation:** Communication strategies will adapt to the available mediums and device resources without centralized control. Communications will continue despite disruption of individual protocols through adaptive selection and opportunistic protocol switching. Protocol 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 [31] – 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, lightweight data compression and reduction techniques will be employed to make the most out of the limited storage and battery available, and adaptive protocol selection will take into account device capabilities and available resources.

### Non-Functional Requirements

**Optional Cloud Integration:** When infrastructure *is* available, it should be taken advantage of. Thus,  $\mu$ -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 System Assumptions and Threat Model

$\mu$ -Babel’s approach to security will reflect the priorities of disaster-scenario deployments, prioritizing resilience over defending against all possible attacks. We establish the following assumptions and threat boundaries:

"something  
something  
these are not  
essential ..." ou  
o parágrafo an-  
terior basta?

passar isto  
para depois do  
system model?

### 3.3.1 Operational Assumptions

**Honest Majority:** Most nodes behave correctly and in accordance with established protocol specifications. We assume adversaries lack the coordination capabilities or resources to simultaneously compromise a majority of the network.

**Controlled Provisioning:** Cryptographic primitives are distributed during initial device installation – before any disasters. This is consistent with planned infrastructure deployments where sensors are provisioned as part of their setup.

worth including sequer?

**Local Adversaries:** Attackers can monitor or disrupt communication within local regions but cannot coordinate simultaneous attacks across the entire network. This reflects the physical constraints of Radio frequency (RF) jamming and the distributed nature of disaster scenarios.

### 3.3.2 Threat Model

A sophisticated adversary capable of large-scale infrastructure compromise would target communication systems regardless of deployed security mechanisms, so our threat model prioritizes resilience against infrastructure *failure* while providing reasonable security against opportunistic attacks during disasters.

We consider adversaries with the following capabilities, and briefly present our corresponding defenses:

- **Passive observation:** Can monitor wireless communications within radio range. Advanced Encryption Standard (AES) encryption prevents plaintext exposure.
- **Active interference:** Can disrupt specific protocols or channels locally. Multi-protocol diversity enables fallback to available mediums.
- **Node compromise:** Can physically remove devices from the network (denial of service). Flash encryption prevents secret extraction from captured devices.
- **Message injection:** Can attempt to inject forged messages without valid credentials. Message Authentication Code (MAC) verification detects and rejects unauthorized transmissions.
- **Replay attacks:** Capture and retransmit previously valid messages. Sequence numbers or timestamp-based checks can detect and reject stale messages.

### 3.3.3 Explicitly Out of Scope

**TODO:** meter algo aqui ou not really? byzantine attacks, etc?

## 3.4 Hardware Platform

**$\mu\text{-Babel}$**  targets resource-constrained embedded platforms commonly used in IoT and domotics systems. The following sections cover device specifications and communication protocol characteristics, establishing the foundation for our multi-protocol, heterogeneous architecture.

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

**Class-1: Battery-Powered Sensor Nodes:** The primary targets for  **$\mu\text{-Babel}$** . These embedded devices handle sensing, actuation, P2P communication, and decentralized coordination without requiring external infrastructure.  **$\mu\text{-Babel}$**  operates autonomously with Class-1 devices alone.

**Class-2: Optional Aggregation Nodes:** Resource-rich devices that can enhance deployments with data aggregation, and optional cloud bridging when infrastructure is available. These devices run the broader Babel [18] framework and are mentioned here for completeness, but are not required for  **$\mu\text{-Babel}$** 's core autonomous operation.

### 3.4.1 Class-1: Sensor Nodes

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.

Table 3.1: Class-1 Device Specifications

Device	CPU <sup>1</sup>	CPU Clock	SRAM	PSRAM <sup>2</sup>	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

<sup>1</sup> All supported devices are 32-bit

<sup>2</sup> 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:

**Group Leader Preference:** Higher-resource devices (ESP32-S3/P4) are prioritized in local Group leader (GL) elections due to their ability to efficiently aggregate member MACs and maintain group state (Section ??). Resource availability merely influences preference, not eligibility.

**Protocol Bridging:** Devices with broader protocol support naturally serve as bridges between protocol domains, forwarding messages across different mediums based on advertised capabilities.

All Class-1 devices support flash encryption for cryptographic primitive storage.

not sure how  
pico handles  
it, tenho de  
confirmar se  
isto é verdade

### Protocol Diversity and Device Coverage

Table 3.2 shows the communication protocols supported across Class-1 devices, demonstrating considerable heterogeneity.

Table 3.2: Supported Communication Protocols

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

<sup>1</sup> Via external SX1276/SX1262 transceiver, controlled through Serial Peripheral Interface (SPI)

<sup>2</sup> ESP32-C5/C6 also support 802.11ax (Wi-Fi 6) and Thread, but we focus on ZigBee given its wider adoption in IoT systems

<sup>3</sup> ESP32-P4 does not possess wireless capabilities by itself and relies on a C6-Mini coprocessor, with ZigBee support still in development

Protocol support across devices provides several advantages:

**Coverage:** All devices support at least three protocols (Wi-Fi, Bluetooth Low Energy (BLE), LoRa), ensuring baseline multi-protocol operation across the entire platform range. Even the least capable device (Pico 2 W) can participate in protocol-diverse mesh networks.

**Protocol-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.

**Long-Range Capabilities:** All platforms support external LoRa transceivers via SPI, enabling long-range communication for disaster scenarios. This addresses the multi-protocol resilience requirements discussed in Section 2.3.

These factors combine to provide multi-protocol 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 device capabilities and protocol characteristics establish the hardware foundation for  **$\mu$ -Babel**'s infrastructure-independent operation. The following section details how these resources are orchestrated through  **$\mu$ -Babel**'s architectural approach.

### 3.4.2 Class-2: Aggregation Nodes

For deployments where additional resources are available, Class-2 devices can enhance  **$\mu$ -Babel** networks without introducing coordinator dependencies. Table 3.3 presents the resources available to these devices.

Table 3.3: Class-2 Device Specifications

Device	CPU <sup>1</sup>	CPU Clock	RAM <sup>1</sup>	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

<sup>1</sup> Configurations with less Random-access memory (RAM) are available but were not considered

Class-2 devices serve as *optional* infrastructure that, when present, can perform:

- Data aggregation and long-term storage
- Computationally intensive analytics beyond Class-1 capabilities
- Optional cloud bridging when external connectivity is available and desired (e.g., MQTT brokerage, CoAP endpoints)

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

## 3.5 Software Stack and Development Environment

**$\mu$ -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.

REWORD  
THIS SINCE IT  
WAS MOVED

### 3.5.1 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, so by using it as a common execution runtime,  **$\mu$ -Babel** maintains a consistent concurrency and timing model across heterogeneous microcontroller platforms, simplifying both portability and implementation.

### 3.5.2 Platform SDKs and Hardware Abstractions

Hardware-specific functionality is accessed through vendor abstractions. The Espressif IoT Development Framework (ESP-IDF) [15] is used for the ESP32 family, while the Raspberry Pi Pico SDK [36] 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  **$\mu$ -Babel** to target multiple microcontrollers through a set of thin Hardware Abstraction Layers (HALs), enabling re-utilization of higher-lever protocols and coordination logic across Class-1 devices.

mencionar anywhere que usamos lwip?  
custom lora drivers, etc?

### 3.5.3 Implementation Language

The  **$\mu$ -Babel** framework is implemented in the C programming language. This is the language of choice for the target development platforms, as it allows precise control over memory usage, interrupt behavior, and timing. These properties are essential in embedded environments with limited memory strict real-time constraints.

### 3.5.4 Concurrency Model

**$\mu$ -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.

An event dispatcher provides higher-level coordination by routing events between protocol components and system services. This structure decouples individual components, simplifies reasoning about system behavior, and supports incremental extension of the protocol stack.

## 3.6 System Model

### 3.6.1 Architecture

The foundational building blocks of the  **$\mu$ -Babel** architecture can be separated into the following *components*, which aim to address each of the aforementioned technical challenges.

#### Mesh Topology Manager

As discussed in Section 2.2, tree topologies and coordinator-based approaches limit resilience, and heavyweight middleware platforms or HyParView's routing and Transmission Control Protocol (TCP) requirements exceed embedded device capabilities.

**$\mu$ -Babel** intends to address these limitations through a fully distributed P2P mesh architecture without coordinator dependencies. The system adapts HyParView's hybrid view concept (maintaining small active neighbor sets for actual communication and larger passive backup lists for failure recovery) but implements discovery and maintenance at the link layer using BLE and LoRa advertisements rather than at the transport layer via TCP connections, accommodating the connectionless nature of embedded radio/wireless communication.

Neighbor discovery is performed through advertising beacons emitted by one or more of the supported protocols (e.g., BLE, LoRa, Wi-Fi) carrying relevant peering information (node identity, capabilities, current neighbor counts, etc.), with periodic gossip exchanges that propagate topology information beyond single-hop range.

It's immediately apparent that – if these exchanges aren't properly integrated – the same node advertising itself through different protocols could lead to duplicate neighbor entries on the receiver side. To address this, nodes possess unique IDs that are consistent across protocols, and by advertising their capabilities they simultaneously inform potential neighbors of which protocols they can expect to receive their advertisements from, which prevents unnecessary processing of packets that would only contain redundant information (e.g., by peeking at the sender's ID). Nodes supporting multiple protocols naturally serve as bridges, enabling topology formation and message forwarding between nodes that might not share the same protocol stack, this is exemplified in Figure ??.

This approach mitigates potential range limitations: while BLE typically provides 10-50m coverage depending on environment and antenna configuration, gossip-based

"temos ainda  
de endereçar o  
wireless"

ADD FIGURE  
WITH (LORA  
ONLY) ->  
(BLE/LORA)  
-> (BLE ONLY)

propagation enables nodes to learn about distant neighbors through multi-hop dissemination, supporting informed routing decisions and topology adaptation.

Unlike the tree topology of Multi-Protocol Gateway or the Routing Protocol for Low-Power and Lossy Networks (RPL) Destination-oriented directed acyclic graph (DODAG) structure of heterogeneous disaster architecture,  **$\mu$ -Babel** will implement true P2P mesh networking where any node can communicate with any other node through dynamically selected multi-hop paths.

Route selection considers multiple factors gathered through local observation and gossip: link quality metrics (Received signal strength indicator (RSSI), Packet loss ratio (PLR)), neighbor availability across different protocols, and current energy budgets.

Store-and-forward mechanisms inspired by the Bundle Protocol will enable operation during network partitions, but will be adapted for P2P rather than cloud-centric operation: data parcels are compressed (using techniques from Section 2.5), queued in local storage (flash, MicroSD cards) when no forward path exists, and opportunistically transmitted when connectivity windows emerge, be it through topology changes or protocol switching creating new communication opportunities.

Coordination emerges from local interactions via gossip-based information propagation, decentralized time synchronization (Section 2.4), and adaptive protocol selection (Section 2.3), rather than centralized allocation or scheduling of roles.

### Multi-Protocol Communication Layer

While existing multi-channel approaches rely on centralized decision-making for channel selection – whether through master nodes (MCSC-WoT), Software-defined networking (SDN) controllers (MINOS), or infrastructure endpoints –  **$\mu$ -Babel** will address these limitations through decentralized *protocol* selection, extending beyond single-medium channel hopping.

The system will employ opportunistic multi-protocol operation (BLE + Wi-Fi + LoRa + ESP-NOW) wherein devices adaptively select communication mediums based on factors like message priority, neighbor availability, energy budget, network conditions and device capabilities. This selection will be performed through local decision-making informed by information gathered through gossip protocols, without requiring SDN controllers or infrastructure coordination.

Hysteresis control mechanisms adapted from the Adaptive Protocol Selection Framework prevent oscillations in these decisions while allowing rapid response to changing conditions.

Protocol-specific compression strategies account for the different energy /bandwidth trade-offs across the proposed stack (e.g. aggressive compression for (relatively) energy-expensive LoRa, lighter compression for short-range BLE), integrating the receiver energy asymmetry observations into power management decisions.

Unlike AWCT’s LoRa Wide Area Network (LoRaWAN) gateway dependency, Heterogeneous IoT’s Near Vertical Incidence Skywave (NVIS) centralized backhaul, MCSC-WoT’s master-based synchronization, or MINOS’s SDN controller requirement,  **$\mu$ -Babel** operates autonomously in a P2P manner with no static coordinator dependencies.

The multi-protocol capability provides resilience through diversity rather than optimization through centralized selection: when conditions render one medium unsuitable, devices autonomously transition to alternatives without requiring coordinator intervention.

### Decentralized Synchronization Service

While MCSC-WoT demonstrates master-based synchronization feasibility and Randomized gossip-consensus-based sync (RGCS) provides masterless convergence, both assume homogeneous single-protocol networks.

**$\mu$ -Babel** will extend RGCS to heterogeneous multi-protocol networks through protocol-specific Poisson processes with distinct activation rates. Rather than a single  $\lambda$  controlling overall sync frequency, the system employs per-protocol gossip rates that take into account the energy required for each transmission, and the current network conditions:

**$\lambda_{BLE}$**  (high frequency): provides rapid local synchronization with nearby neighbors, leveraging BLE’s low energy cost for frequent exchanges

**$\lambda_{LoRa}$**  (low frequency): maintains long-range temporal coordination, using LoRa’s more expensive transmissions sparingly

**$\lambda_{Wi-Fi} / Others$**  (medium frequency): situational use based on network density, energy budgets and other factors

Each protocol will have its own Poisson-triggered gossip loop, but all updates contribute to a single shared logical clock via the converge-to-max criterion explored above. When sync events from different protocols occur near-simultaneously, the max operation’s associativity ensures correct convergence regardless of update order.

This multi-graph gossip approach provides several advantages over homogeneous sync:

**Faster convergence:** High-frequency BLE gossip can achieve tighter local consensus while sparse LoRa exchanges prevent drift between distant clusters. The combination converges faster than either protocol alone.

**Resilience through diversity:** If 2.4 GHz interference disrupts BLE/Wi-Fi synchronization, LoRa maintains loose temporal coordination across the network. Conversely, if LoRa experiences poor conditions, local BLE sync keeps nearby nodes coordinated.

**Natural energy optimization:** Protocol characteristics directly inform sync frequencies, so that more expensive long-range transmissions occur rarely while cheap short-range exchanges happen frequently, matching energy budgets to communication needs.

The hysteresis control mechanisms adapted from Section 2.3 will prevent oscillations in protocol selection during sync events: sync partner and protocol choices stabilize around locally optimal configurations rather than continuously switching between equally-viable options.

The approach aims to handle network partitions gracefully: nodes maintain synchronization within their reachable partition using available protocols, and when partitions merge (through mobility or topology changes), the converge-to-max criterion naturally reconciles previously independent temporal references without requiring special merge logic.

Unlike MCSC-WoT's master-based approach, this architecture will operate in a fully P2P fashion with no fixed coordinator dependencies. The multi-protocol extension goes beyond RGCS's original homogeneous network assumptions, since heterogeneous gossip graphs with protocol-specific characteristics warrant distinct sync frequencies, enabling simultaneous optimization of convergence speed, energy consumption, and range coverage.

## Data Reduction and Compression Module

Extending Ambrosia's lightweight prediction approach to multi-protocol scenarios,  $\mu$ -Babel will reduce data transmission through prediction-based filtering: nodes transmit sensor readings only when prediction error exceeds configurable thresholds (similar to Ambrosia), with protocol-aware tuning to balance accuracy against energy constraints across heterogeneous protocols.

maybe not  
worth defining  
as a ""component"" per se

Building on prediction-based reduction, the goal is to apply Delta+Run-length encoding (RLE) compression to the readings that are transmitted, combining both techniques for added energy savings and reduced data volume over the air.

Following the two-tier architecture pattern, compression occurs at sensor nodes before transmission, with just the first sample in each window being transmitted with no compression applied.

Adaptive threshold selection will adjust  $\delta$  based on both channel characteristics and node state:

- **High  $\delta$  for LoRa:** Tolerates larger prediction errors to minimize expensive long-range transmissions, sending only when predictions deviate significantly. These packets undergo aggressive Delta+RLE compression to minimize airtime costs.

estes deltas  
vinham do am-  
brosia, fica con-  
fuso agora que  
falo de delta  
encoding?

- **Low  $\delta$  for BLE:** Requires tighter prediction accuracy for cheap local exchanges, allowing for more frequent transmissions to maintain precision. Lighter compression (delta encoding only) is applied here for lower latency.
- **Battery-aware adaptation:** As node energy depletes, increased  $\delta$  across all channels reduces transmission frequency, extending lifetime for critical messages. Compression at this point becomes more lossy to further minimize transmitted data volume.

**Application-aware transmission** will distinguish between data types: periodic sensor readings (temperature, humidity, light) use window-based prediction with configurable  $\delta$  thresholds, while discrete events (emergency alerts, button presses, motion detection) transmit immediately without prediction. For predictable streams, accuracy-critical applications maintain low  $\delta$ , while trend monitoring can accept higher values to reduce transmission frequency.

Compression strategies also vary by data type: environmental sensors can tolerate more prominent Delta+RLE compression given their high temporal correlation, while critical alerts transmit with minimal (if any) compression to reduce latency.

**Gateway energy management** accounts for the 15-20% receiver energy cost by implementing selective radio shutdown: gateways can disable specific protocols during low-activity periods, waking periodically to check for incoming sync beacons or when prompted via other protocols. Decompression overhead in these nodes is minimal compared to transmission costs at the sensor level.

The expectation is that the combination of lightweight prediction, adaptive thresholding, and protocol-specific tuning together with Delta+RLE compression will provide substantial energy savings without requiring complex and computationally intensive compression algorithms that would themselves consume significant power.

entrar pela  
coisa de desli-  
gar os rádios  
ainda mais?

# 4

## PROGRESS REPORT AND PLANNING

### 4.1 Progress Report

### 4.2 Planning

#### 4.2.1 Calendarization – GANTT

## 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](https://doi.org/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](https://doi.org/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. 33).
- [5] AWS open source. *Intertask Communications*. 2026. url: [https://www.freertos.org/message\\_passing\\_performance](https://www.freertos.org/message_passing_performance) (visited on 2026-01-27) (cit. on p. 33).
- [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. 15–18, 23).
- [7] D. Blalock, S. Madden, and J. Guttag. "Sprintz: Time Series Compression for the Internet of Things". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2.3 (2018-09). doi: [10.1145/3264903](https://doi.org/10.1145/3264903) (cit. on pp. 21, 22).
- [8] 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](https://doi.org/10.1109/MIC.2012.29) (cit. on p. 8).
- [9] 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](https://doi.org/10.1109/SCC.2019.00019) (cit. on pp. 12, 13).

- 
- [10] 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).
  - [11] 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](https://doi.org/10.1109/JIOT.2023.3283347) (cit. on p. 23).
  - [12] 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](https://doi.org/10.3390/s24165320) (cit. on p. 1).
  - [13] C. J. Deepu, C.-H. Heng, and Y. Lian. "A Hybrid Data Compression Scheme for Power Reduction in Wireless Sensors for IoT". In: *IEEE Transactions on Biomedical Circuits and Systems* 11.2 (2017), pp. 245–254. DOI: [10.1109/TBCAS.2016.2591923](https://doi.org/10.1109/TBCAS.2016.2591923) (cit. on pp. 20, 22).
  - [14] 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](https://doi.org/10.1109/NGMAST.2015.71) (cit. on p. 6).
  - [15] 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. 33).
  - [16] 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](https://doi.org/10.1007/s10796-024-10496-9) (cit. on p. 6).
  - [17] 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](https://doi.org/10.1007/978-3-642-36958-2_13) (cit. on p. 23).
  - [18] 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](https://doi.org/10.1109/SRDS55811.2022.00022) (cit. on pp. 3, 30).
  - [19] 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](https://doi.org/10.1145/3290605.3300498) (cit. on p. 6).
  - [20] 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](https://doi.org/10.1109/JIOT.2018.2846299) (cit. on p. 23).

## BIBLIOGRAPHY

---

- [21] 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](https://doi.org/10.3390/s20226441) (cit. on p. 1).
- [22] 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](https://doi.org/10.1109/TPDS.2012.50) (cit. on p. 24).
- [23] 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](https://doi.org/10.1109/ICTEmSys.2019.8695968) (cit. on pp. 10, 13, 15, 16).
- [24] 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).
- [25] 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](https://doi.org/10.1109/DSN.2007.56) (cit. on pp. 10, 13, 18).
- [26] 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/ICPADS47876.2019.90087](https://doi.org/10.1109/ICPADS47876.2019.90087) (cit. on pp. 14, 16).
- [27] 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](https://doi.org/10.1109/SSIRI.2008.30) (cit. on p. 23).
- [28] 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).
- [29] 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).
- [30] 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](https://doi.org/10.1109/INFCOMW.2018.8406884) (cit. on pp. 12, 13).
- [31] J. Moy. *OSPF Version 2*. RFC 1247. 1991-07. DOI: [10.17487/RFC1247](https://doi.org/10.17487/RFC1247) (cit. on p. 28).

- 
- [32] 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](https://doi.org/10.1109/JIOT.2022.3204335) (cit. on p. 23).
  - [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](https://doi.org/10.1109/JIOT.2017.2767608) (cit. on p. 2).
  - [34] J. Porte et al. "Heterogeneous wireless IoT architecture for natural disaster monitoring". In: *EURASIP Journal on Wireless Communications and Networking* 2020.1 (2020), p. 184 (cit. on pp. 11, 13, 14, 16).
  - [35] A. K. M. Al-Qurabat, C. Abou Jaoude, and A. K. Idrees. "Two Tier Data Reduction Technique for Reducing Data Transmission in IoT Sensors". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 2019, pp. 168–173. doi: [10.1109/IWCMC.2019.8766590](https://doi.org/10.1109/IWCMC.2019.8766590) (cit. on pp. 21, 22).
  - [36] Raspberry Pi Ltd. *Raspberry Pi Pico SDK*. 2026. url: [https://www.raspberrypi.com/documentation/pico-sdk/index\\_doxxygen.html](https://www.raspberrypi.com/documentation/pico-sdk/index_doxxygen.html) (visited on 2026-01-27) (cit. on p. 33).
  - [37] 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](https://doi.org/10.1109/TVT.2019.2904244) (cit. on pp. 1, 2).
  - [38] K. Scott and S. C. Burleigh. *Bundle Protocol Specification*. RFC 5050. 2007-11. doi: [10.17487/RFC5050](https://doi.org/10.17487/RFC5050) (cit. on p. 12).
  - [39] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. 2014-06. doi: [10.17487/RFC7252](https://doi.org/10.17487/RFC7252) (cit. on p. 8).
  - [40] R. H. Shumway and D. S. Stoffer. "ARIMA Models". In: *Time Series Analysis and Its Applications: With R Examples*. Cham: Springer International Publishing, 2017, pp. 75–163. isbn: 978-3-319-52452-8. doi: [10.1007/978-3-319-52452-8\\_3](https://doi.org/10.1007/978-3-319-52452-8_3) (cit. on p. 20).
  - [41] 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).
  - [42] 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).
  - [43] S. Suryavansh et al. "A data-driven approach to increasing the lifetime of IoT sensor nodes". In: *Scientific Reports* 11.1 (2021), p. 22459. doi: [10.1155/2018/4283087](https://doi.org/10.1155/2018/4283087) (cit. on pp. 19, 22).

- [44] 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, 26).
- [45] 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](https://doi.org/10.1109/MCOM.001.1900056) (cit. on pp. 15, 16).
- [46] 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](https://doi.org/10.1109/Blockchain.2019.00031) (cit. on p. 24).
- [47] 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. 11).
- [48] 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](https://doi.org/10.1109/MILCOM.2018.8599710) (cit. on p. 24).
- [49] 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](https://doi.org/10.1109/JIOT.2020.2993216) (cit. on p. 2).
- [50] 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](https://doi.org/10.1155/2018/4283087) (cit. on pp. 17, 18).
- [51] 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](https://doi.org/10.1109/JIOT.2020.3032757) (cit. on p. 23).
- [52] 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](https://doi.org/10.1109/ACCESS.2017.2778504) (cit. on p. 1).
- [53] D. Žatuchin and M. Azarskov. “An Adaptive Protocol Selection Framework for Energy-Efficient IoT Communication: Dynamic Optimization Through Context-Aware Decision Making”. In: *Informatics* 12.4 (2025). issn: 2227-9709. doi: [10.3390/informatics12040125](https://doi.org/10.3390/informatics12040125) (cit. on p. 22).

