Diss. ETH No. 24898

An Efficient Platform and Communication Architecture for Event-triggered Cyber-physical Systems

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

FELIX ERNST SUTTON
MSc ETH Zurich
BE(Hons) & BMaCompSc, The University of Adelaide

born April 3, 1982
citizen of Australia and Switzerland

accepted on the recommendation of
Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Koen Langendoen, co-examiner
Dr. Jan Beutel, co-examiner

2018

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

TIK-SCHRIFTENREIHE NR. 174

Felix Ernst Sutton

An Efficient Platform and Communication Architecture for Event-triggered Cyber-physical Systems

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A dissertation submitted to
ETH Zurich
for the degree of Doctor of Sciences

Diss. ETH No. 24898

Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Koen Langendoen, co-examiner
Dr. Jan Beutel, co-examiner

Examination date: February 9, 2018

To my parents.

Abstract

The interaction of modern cyber technologies with the physical world is widely anticipated to bring significant societal and economic benefits. These cyber-physical systems combine computation and communication technologies to monitor and control complex physical processes, and therefore make it possible to develop smart products and services in industries such as manufacturing, healthcare and agriculture.

The dynamics of many physical processes are inherently unpredictable. This implies that cyber-physical systems must be responsive to non-deterministic events, while remaining energy-efficient for long-term operation and adaptable to the dynamics of the physical process under observation. However, these stringent requirements make it difficult to realize efficient event-triggered cyber-physical systems in practice.

In order to overcome this practical limitation, this thesis presents a new platform and communication architecture for the construction of efficient event-triggered cyber-physical systems with respect to responsiveness, energy-efficiency and adaptability. We make four main contributions:

- We introduce a new platform architecture for composing energy-efficient wireless embedded platforms using a time-predictable processor interconnect called Bolt. The design and prototype implementation of Bolt is presented, which exhibits a throughput of up to 3.3 Mbps and a negligible power overhead of 1.3 μW during periods of inactivity.

- We propose a novel system design methodology for realizing event-triggered cyber-physical systems, subject to responsiveness, energy efficiency and adaptability design constraints. A prototype of a wireless acoustic emission sensing system is presented, which demonstrates significant improvements to the state-of-the-art with respect to the responsive and energy-efficient detection of acoustic events and their multi-hop dissemination.

- We design and implement Zippy, an asynchronous flooding primitive for the rapid dissemination of rare events through a multi-hop network with unprecedented energy-efficiency. The developed prototype exhibits a dissemination latency as low as 24.4 ms for an 8-bit event packet through a 3-hop network, and a power dissipation of 9.6 μW during periods of inactivity.

- We introduce Blitz, the first communication architecture that combines asynchronous and synchronous flooding primitives to facilitate low latency and energy-efficient multi-hop dissemination of events having adaptable bandwidth requirements. Blitz also incorporates a novel scheme for mitigating erroneous wake-ups, which is shown experimentally to reduce energy consumption. Experiments on a prototype of Blitz show a mean latency as low as 108.9 ms for an 8-bit event packet and its associated data packet of 32 bytes through a 4-hop network, and a power dissipation of 16 μW during periods of inactivity.

Zusammenfassung

Die Interaktion von modernen Cyber-Technologien mit der realen Welt wird erwartungsgemäss erhebliche soziale und wirtschaftliche Vorteile bringen. Diese Cyber-Physikalischen-Systeme kombinieren Berechnungs- und Kommunikationstechnologien, um komplexe physikalische Prozesse zu überwachen und zu kontrollieren. Diese Systeme ermöglichen es, intelligente Produkte und Dienstleistungen in den Bereichen Fabrikation, Gesundheitswesen und Landwirtschaft zu entwickeln.

Die Dynamiken vieler physikalischer Prozesse sind naturgemäss unvorhersehbar. Deshalb müssen Cyber-Physikalische-Systeme während des Betriebs auf nicht-deterministische Ereignisse reagieren können sowie einen energieeffizienten Langzeitbetrieb und Anpassungsfähigkeit an die Dynamiken des beobachteten physikalischen Prozesses gewährleisten. In der Praxis erschweren diese Anforderungen die Realisierung von effizienten, ereignisgesteuerten Cyber-Physikalischen-Systemen erheblich.

Um die genannten praktischen Einschränkungen zu überwinden, präsentiert diese Dissertation eine neue Plattform- und Kommunikationsarchitektur für die Konstruktion von effizienten, ereignisgesteuerten Cyber-Physikalischen-Systemen in Bezug auf Reaktivität, Energieeffizienz und Adaptabilität. Wir leisten vier Hauptbeiträge zum Forschungsgebiet Cyber-Physikalische-Systeme:

- Wir stellen eine neue Plattformarchitektur für die Erstellung von energieeffizienten, drahtlosen und eingebetteten Systemen vor, welche auf einem deterministischen Prozessor-Bus namens Bolt basiert. Das Design und die prototypische Implementierung von Bolt wird gezeigt. Bolt ermöglicht einen Durchsatz von bis zu 3.3 Mbps, und weist während inaktiven Perioden einen vernachlässigbaren Energieaufwand von 1.3 μW auf.

- Wir präsentieren eine neue System-Design-Methodologie für die Realisierung von ereignisgesteuerten Cyber-Physikalischen-Systemen, welche strengen Einschränkungen bezüglich Reaktivität, Energieeffizienz und Anpassungsfähigkeit unterliegen. Dazu wird ein Prototyp eines drahtlosen, akustischen Emmissionsvermessungssystems vorgestellt, anhand welchem

deutliche Verbesserungen hinsichtlich der reaktiven und energieeffizienten Erkennung von akustischen Ereignissen sowie deren Multi-Hop-Verbreitung aufgezeigt werden.

- Wir konzipieren und implementieren Zippy, eine asynchrone Flutungsfunktion, welche die rasche Verbreitung von seltenen Ereignissen in einem Multi-Hop-Netzwerk mit einer noch nie dagewesenen Energieeffizienz abwickelt. Der entwickelte Prototyp verfügt über eine Verbreitungslatenz von nur 24.4 ms, um ein 8-bit Ereignispaket in einem 3-Hop-Netzwerk zu verteilen, sowie über eine Verlustleistung von 9.6 μW während Perioden der Inaktivität.

- Wir präsentieren Blitz, die erste Kommunikationsarchitektur, welche asynchrone und synchrone Flutungsfunktionen kombiniert, um geringe Latenzen und eine energieeffiziente Multi-Hop-Dissemination von Ereignissen mit adaptierbaren Bandbreitenanforderungen zu ermöglichen. Blitz umfasst auch ein neues Schema zur Eindämmung von fehlerhaften Wecksignalen, was den Energieverbrauch verringert, wie experimentell erprobt wird. Experimente mit einem Prototypen zeigen eine niedrige Latenz von nur 108.9 ms für die Verbreitung eines 8-bit Ereignispakets und das damit verbundene Datenpaket von 32 Bytes durch ein 4-Hop-Netzwerk, und eine Verlustleistung von 16 μW während Inaktivitätsphasen.

Acknowledgments

I would like to express my gratitude to Prof. Dr. Lothar Thiele for giving me the opportunity to conduct research in his institute, and for the exceptional guidance and support offered to me throughout my doctoral studies. I sincerely thank Prof. Dr. Koen Langendoen and Dr. Jan Beutel for their willingness to review this thesis and to serve on my examination committee.

I have had the pleasure to collaborate with many talented researchers and engineers throughout my research. My special thanks to Reto Da Forno, Bernhard Buchli, Marco Zimmerling, Federico Ferrari, Roman Lim, Tonio Gsell, Georgia Giannopoulou, Samuel Weber, Balz Maag, Lukas Sigrist, Andres Gomez, Olga Saukh, Matthias Meyer, Romain Jacob and David Gschwend. I am also thankful to friends who have offered me great support throughout my studies, in particular Sally and Rico Achenbach and Judith Fries.

Last but not least, I would like to express my deepest gratitude to my family in Australia and Switzerland, in particular Susanne and Adrian Sutton, Michelle and Dominik Sutton, Jolanda and Heini Bossert, and Anita and Hansjörg Bossert, for their unrelenting support. This work would not have been possible to complete without the amazing support of my wife, Katharina Bossert.

The work presented in this thesis was scientifically evaluated by the SNSF and financed by the Swiss Confederation and by Nano-Tera.ch. This support is gratefully acknowledged.

Contents

List of Figures

List of Tables

1 Introduction

It is widely anticipated that the interaction of modern *cyber* technologies, i.e., computation and communication, with the *physical* world, will bring significant societal and economic benefits [RLSS10]. These *cyber-physical systems* promise automated monitoring and control of complex physical processes that govern a wide range of application domains including manufacturing, healthcare, agriculture, logistics and transportation [Lee08]. Consequently, cyber-physical systems are key technological enablers for the Industrial Internet of Things, also referred to as Industry 4.0, which is expected to revolutionize industrial automation toward smart products, smart production and smart services [WSJ17].

The definition of a cyber-physical system is concisely summarized by Lee et al. [LS16] as follows:

> *Cyber-physical systems are heterogeneous blends by nature. They combine computation, communication, and physical dynamics.*

It is evident that these systems are constructed from many different types of components that combine to follow, and possibly influence, the dynamics of physical processes under observation. Computation is required to sense the state of the physical process, characterize the dynamics and determine appropriate actions, while communication is needed to collect the state of the physical process with high-spatial coverage and to close the feedback loop for automated control. While the communication between computing entities of cyber-physical systems may be wired, we focus on wireless technologies due to the potential for high spatial coverage without the high cost and cumbersome deployment associated with cabled systems.

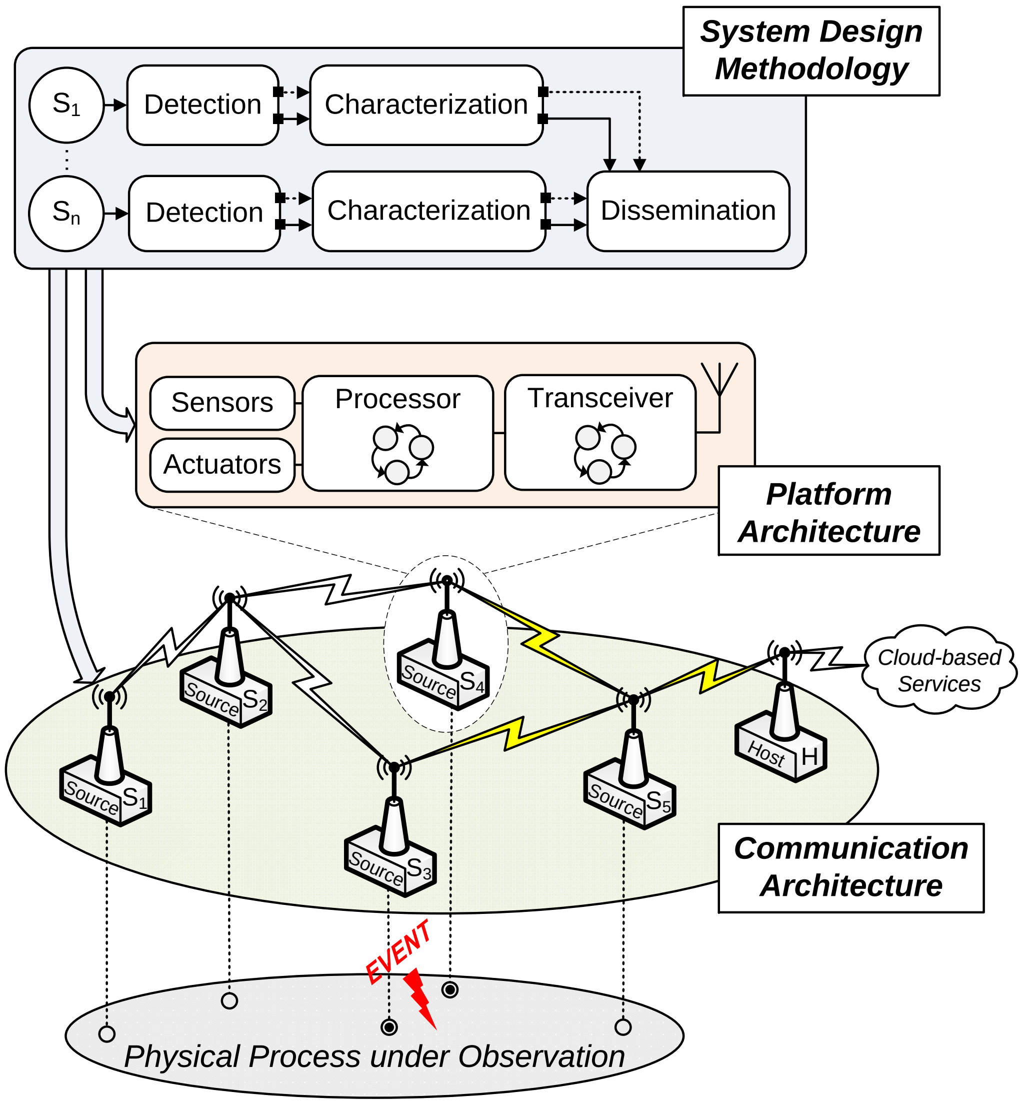


Figure 1.1: An overview of an event-triggered cyber-physical system, where a network of wireless embedded platforms monitors and controls a physical process. The platform architecture defines how the wireless embedded platforms are constructed, the communication architecture defines how platforms share information, and the system design methodology assists in the transformation of functional requirements of the system into an efficient implementation.

The realization of cyber-physical systems is, at least in part, a consequence of the past 15 years of research in the field of wireless sensor networks. This multi-disciplinary research community has produced an extensive collection of theoretical concepts and real-world deployments, as survey in [KW07, Amm14, OBR14], that relate to the sensing of physical processes coupled with wireless communication. The wireless sensing systems considered are typically *periodic*, in that the computation and communication entities are activated periodically according to a specific duty cycle. However, the dynamics of many physical processes do not change periodically, but instead evolve in a non-deterministic manner. Therefore, in order to capture and possibly control these physical processes while conserving precious energy resources, the computation and communication of these cyber-physical systems must be *event-triggered*. That is, these cyber-physical systems monitor and react *on-demand* to non-deterministic events produced by the physical process.

This thesis focuses on event-triggered cyber-physical systems using wireless communication, as depicted in Figure 1.1, where the goal of the system is to monitor and possibly control the dynamics of a physical process. A network of wireless embedded platforms, referred to as source nodes S_i, are responsible for detecting, characterizing and disseminating events to a host H for analysis. The host serves as a gateway to cloud-based services, which may also be used for application-specific event analysis. Each source node incorporates sensors to detect events, a processor to perform application-specific computations, a transceiver to communicate, and actuators to control the state of the physical process as instructed by the host.

When the state of the physical process under observation changes, an event is produced. Depending on the topology of the network, one or more source nodes detect the event, e.g., source nodes S_3 and S_4 as depicted in Figure 1.1. Once an event is detected, the source node collects additional information on the event so to determine its specific properties. The event, together with its characterization, are then disseminated through the multi-hop network to the host where application-specific analysis is performed. The host, possibly in conjunction with cloud-based services, decides on the appropriate response and instructs one or more source nodes to take action. For example, an event may be deemed inconsequential and all source nodes are instructed to return to an energy-saving mode, source nodes may be instructed to use additional sensing modalities to better characterize the event, or actuators may be activated in order to control the behavior of the physical process under observation.

The construction of an event-triggered cyber-physical system is defined by a *platform and communication architecture*, as represented in Figure 1.1. The platform architecture defines how each node is composed, i.e., how sensors, a processor, a transceiver and actuators are interconnected, while the communication architecture defines how a network of source nodes disseminate events and their associated data to the host over a wireless channel. Equipped with an platform and communication architecture, a *system design methodology* assists the system designer in transforming the functional requirements of an event-triggered cyber-physical system into an efficient implementation.

We next detail the system requirements of event-triggered cyber-physical systems, before detailing the specific challenges associated with achieving the desired requirements of the system. We then present an overview of the thesis contributions and how they address the challenges of realizing efficient event-triggered cyber-physical systems.

1.1 System Requirements

In order to realize the full potential of event-triggered cyber-physical systems, it is imperative that the following requirements are satisfied:

- In order to react quickly to the unpredictable dynamics of the physical process under observation, event-triggered cyber-physical systems must be ***responsive***. The platform architecture must therefore support rapid detection and characterization of events, while the communication architecture must facilitate low-latency multi-hop dissemination from one or more source nodes to the host.

- Since wireless embedded platforms are typically battery powered, and may not necessarily have the opportunity for energy harvesting, event-triggered cyber-physical systems must be ***energy-efficient***. This requires that both platform and communication architecture minimize energy consumption whenever possible in order to maximize the operational lifetime of the system.

- Due to the application-specific nature of wireless embedded platforms, a platform architecture must promote independent design and component-level re-use. Therefore, the platform architecture must facilitate the ***composable*** construction of wireless embedded platforms without adversely impacting responsiveness and energy efficiency.

- Given the inherent heterogeneity of physical processes, the event-triggered cyber-physical system must be ***adaptable*** to changes in the environment. The platform and communication architectures must support on-demand resource and configuration updates, such as the on-demand activation of multi-modal sensors on multiple source nodes or the dynamic bandwidth allocation in response to a detected event.

- In an effort to simultaneously support responsiveness, energy-efficiency, composability and adaptability, a systematic ***system design methodology*** is needed to assist the system designer in transforming application-specific requirements into an efficient implementation.

1.2 Challenges

System Design Methodology. The design and implementation of event-triggered cyber-physical systems is difficult in practice due to the inherent complexity and the conflicting system requirements.

Event-triggered cyber-physical systems consist of a complex combination of analog, digital and software components. This system-wide complexity makes it difficult for a system designer to comprehend all component-level dependencies and how their unique trade-offs influence the performance of the overall system. For example, selecting a component that exhibits a low power dissipation at the cost of a long processing delay may satisfy the component-level requirements of energy efficiency, but based on the component dependencies, this may result in an increase of latency, thus violating the system requirement of responsiveness.

The system requirements of responsiveness, energy-efficiency and adaptability are in conflict with each other. To give a concrete example, we consider two approaches to designing an event-triggered cyber-physical system. The first is an always-on approach, where all components are active irrespective of whether an event is detected or not. The resulting system is indeed responsive and adaptable, as all components are always on they exhibit a minimal response time, and consequently can be reconfigured at any moment. However, the system is not energy-efficient, as significant energy resources are consumed regardless of whether an event is produced by the physical process under observation. An alternative is to use a duty-cycled/on-demand approach, where components of the system are only activated periodically or on-demand. The energy-efficiency of the system improves significantly, but adversely impacts responsiveness and adaptability, since components may not be active when an event or a configuration change occurs, or the time to activate components increases latency.

This combination of complexity and conflicting system requirements leads to an interesting dilemma for the system designer. On the one hand, the complex component dependencies make it difficult to perform appropriate component-level trade-offs, but on the other hand, performing these component-level trade-offs is the only way of balancing the conflicting system requirements. Therefore, there is a need for a systematic system design methodology that can introduce structure and use levels of abstraction to guide the system designer through these delicate component-level trade-offs and realize efficient event-triggered cyber-physical systems.

Platform Architecture. The typical wireless embedded platform used for wireless sensor network deployments consists of a single processor interfaced to sensors, actuators and a transceiver [SKK+12]. However, the increased processing demands on computation coupled with stringent real-time constraints on communication introduces resource interference. This means that the interleaved execution of sensing, actuation and communication tasks encounter resource interference with respect to time, power and clock domains.

This resource interference adversely impacts the performance of the platform in terms of responsiveness and energy-efficiency. For example, if an event is detected while the communication task is executing, the respective sensing and communication tasks will compete for resources, such as clock cycles, memory and peripherals. In such a scenario, either the delayed execution of the sensing task results in a delayed or missed event detection, or the stalled execution of the communication task impacts the reliability, and possibly the energy efficiency, of the communication protocol. Irrespective of the approach taken by high-level software constructs, resource interference adversely impacts the responsiveness and energy efficiency of the system.

The emerging trend toward heterogeneous multi-processor platforms fails to circumvent the aforementioned resource interference problem. While shared busses or shared memory facilitate the transfer of information between processors, they still cause resource interference in the time, power and clock domains. Therefore, there is a need for a new platform architecture that takes advantage of processor heterogeneity and facilitates bi-directional information transfer between processors, while avoiding resource interference. As a consequence, application-specific wireless embedded platforms can be realized by composing independent computation and communication components, thereby promoting independent design and component-level re-use.

Communication Architecture. The past decade and a half of wireless sensor network research has produced a variety of energy-efficient multi-hop wireless communication protocols based on periodic communication, as surveyed in [HXS+13]. Multi-hop dissemination is achieved by coordinating nodes in the network to periodically communicate with their neighbors according to a duty-cycle, which may be statically defined at each node or dynamically initialized using network-wide synchronization techniques.

While this well-studied class of communication protocols has been successfully deployed in the field for multiple years with high reliability and energy-efficiency [BBF+11], this protocol class does however exhibit an undesirable trade-off between responsiveness and energy-efficiency.

That is, in the context of an event-triggered cyber-physical system, in order to reduce the latency of disseminating an event from any source node to the host, the duty-cycle of all nodes in the network must be increased, which results in an increase in energy consumption. In application scenarios where events are deterministic, these radio duty-cycled protocols can, at least in principle, be tuned to the periodicity of the dynamics of the physical processes under observation, and therefore provide low-latency dissemination without superfluous communication. However, when events occur non-deterministically, as is the case for event-triggered cyber-physical systems, significant energy is wasted as periodic communication proceeds regardless of an event being detected.

In order to provide responsive and energy-efficient dissemination of non-deterministic events, the undesirable trade-off between latency and energy efficiency exhibited by modern duty-cycled protocols must be circumvented. Therefore, there is a need for a new communication architecture that supports on-demand multi-hop communication that exhibits both low latency and energy efficiency.

1.3 Thesis Outline and Contributions

In this thesis, we present a platform and communication architecture for the systematic design of efficient event-triggered cyber-physical systems. The main contributions are the following:

Chapter 2: Composable and Energy-efficient Wireless Embedded Platforms. We address the challenges of resource interference by proposing a new platform architecture that supports composable construction of energy-efficient wireless embedded platforms.

The proposed architecture separates computation and communication tasks onto dedicated processors. Information transfer between the two processors is facilitated by asynchronous message passing using Bolt, a processor interconnect with predictable run-time behavior. Predictable information transfer between processors is achieved by tightly bounding the worst-case execution time of Bolt message passing, as verified by formal methods, while decoupling the processors with respect to time, power and clock domains. The predictable run-time behavior of Bolt facilitates the composable construction of heterogeneous wireless embedded platforms, thereby promoting independent design and component-level re-use.

We present the design and prototype implementation of Bolt, which supports a message throughput of up to 3.3 Mbps and a negligible power overhead of 1.3 μW during periods of inactivity.

Chapter 3: Design Methodology for Efficient Event-triggered Wireless Sensing Systems. We manage the inherent complexity and evaluate component-level trade-offs by utilizing a combination of well-established design principles together with a design guideline. The proposed four step design process guides the system designer toward an efficient realization of an event-triggered cyber-physical system subject to responsiveness, energy efficiency and adaptability design constraints.

The functionality of an event-triggered cyber-physical systems is first partitioned into a pipeline of logical event-triggered components. Each component is then represented by a logical model that encapsulates responsiveness, energy efficiency and adaptability design constraints, and adheres to an event-triggered interface specification. The design and implementation of each logical component is then performed in isolation using well-known design techniques with an appropriate design guideline. All components are then integrated onto a physical platform architecture, as presented in Chapter 2, without compromising the underlying design constraints.

We demonstrate the benefits of the proposed methodology by realizing a wireless acoustic emission sensing system, consisting of an acoustic sensor interface, an event characterization component and an event-based synchronous protocol termed the eLWB for multi-hop event dissemination. Experiments show that the prototype is capable of detecting an acoustic event with a delay of 16 μs, starting to characterize an acoustic event with a delay of 29 μs, and disseminating an acoustic event and its associated data over a multi-hop network with an average latency as low as 113.2 ms, while dissipating a total of 59.7 μW during periods of inactivity.

Chapter 4: On-demand Network Flooding. We demonstrate the first step toward low-latency and energy-efficient multi-hop communication by circumventing the fundamental trade-off exhibited by modern radio duty-cycled protocols. This is achieved through the design and implementation of an asynchronous flooding primitive for the rapid dissemination of rare events through a multi-hop network, which dissipates less than 10 μW during periods of inactivity.

We present Zippy, an on-demand flooding protocol based on low-complexity radio hardware. Zippy takes advantage of the ultra-low power dissipation and unique timing properties of low-complexity receivers to achieve asynchronous network wake-up, fine-grained per-hop synchronization, and efficient multi-hop dissemination of a small event packet to the host.

We detail the design, analysis and experimental evaluation of Zippy in a laboratory setting and in an indoor testbed. Experiments show that

Zippy supports on-demand flooding of a small event packet through a multi-hop network with an end-to-end latency as low as 24.4 ms for an 8-bit event packet through a 3-hop network, a per-hop synchronization as low as 21.9 μs, and a power dissipation of 9.6 μW during periods of inactivity.

Chapter 5: Low Latency and Energy-efficient Event-triggered Wireless Communication. We combine the asynchronous network wake-up provided by Zippy, as presented in Chapter 4, with the synchronous event-based dissemination protocol eLWB, as presented in Chapter 3, to achieve multi-hop event dissemination with unprecedented latency and energy efficiency.

We present Blitz, the first communication architecture that combines asynchronous and synchronous communication primitives to rapidly wake-up a multi-hop network, arbitrate network bandwidth, and disseminate events having variable bandwidth requirements on-demand. Blitz takes advantage of ultra-low power wake-up receivers to wake-up all nodes by flooding wake-up preambles, and then proceeds with a synchronous flooding-based protocol for the arbitration of network resources and scheduling of contention-free time-slots for event and data dissemination. In addition, Blitz integrates a novel classification technique for mitigating erroneous wake-ups, a fundamental limitation of ultra-low power wake-up receivers.

Using an analytical model, we show that Blitz is more energy efficient and exhibits a lower worst-case latency compared to the eLWB for a wide range of event-triggered application scenarios. The analysis results also highlights that the combination of asynchronous and synchronous primitives, as used in Blitz, yields superior latency and energy-efficiency compare to the eLWB for both rare and frequent events.

We design and implement a prototype of Blitz by extending the wireless acoustic emission sensing system presented in Chapter 3. We experimentally evaluate the event-triggered cyber-phyical system in laboratory and testbed conditions. Experiments show that the developed prototype supports a mean latency as low as 108.9 ms for an 8-bit event packet and its associated data packet of 32 bytes through a 4-hop network, while dissipating 16 μW, plus an additional 3 μW for the acoustic sensor interface, during periods of inactivity.

2

Composable and Energy-efficient Wireless Embedded Platforms

Increased processing demands and real-time constraints of modern wireless cyber-physical systems are driving a platform paradigm shift from classical single-processor motes toward heterogeneous multi-processor platforms. These emerging platforms promise efficient concurrent processing with energy-proportional system performance. However, the use of shared interconnects and shared memory for inter-processor communication causes interference in the time, power, and clock domains. This prevents system designers from fully harnessing these benefits, since the inherent resource interference adversely impacts the responsiveness and energy efficiency of the platform, while hindering modular design and violating separation of concerns.

In this chapter, we address these platform limitations with Bolt, an ultra-low power processor interconnect that facilitates the composable construction of heterogeneous wireless embedded platforms. We present an architectural blueprint for interconnecting two independent processors, while enabling asynchronous inter-processor communication with predictable run-time behavior. We detail a prototype implementation of Bolt, and apply formal methods to analytically derive bounds on the execution time of its message passing operations. Experiments with a custom-built dual-processor platform demonstrate that the Bolt prototype exhibits predictable message passing with empirical bounds that match the analytical bounds to within a few clock cycles, achieves a high throughput of up to 3.3 Mbps, and incurs a negligible power overhead relative to state-of-the-art platforms.

2.1 Introduction

In the early days of sensor networks, platforms featuring a single 8-bit or 16-bit microcontroller, such as Mica [HC02] and Telos [PSC05], spawned the development of a wide range of sense-and-send applications. These platforms offered modest computing resources well matched to the demands of low-rate sensing of physical parameters such as temperature, humidity, light, etc. Low-power operation was achieved by interleaving sensing, data processing, and communication tasks and by carefully managing the power state of hardware components.

Challenges. Although this design approach has been extensively followed to demonstrate the feasibility of wireless sensing applications, through our own experiences in designing, developing, and maintaining large-scale sensor network installations, we have encountered recurring patterns that impede the construction and performance of wireless embedded systems. We have observed that the engineering effort in realizing such systems is labor-intensive with respect to the design, test, and diagnosis of hardware and software components. While difficult to quantify, we argue that these practical complexities lead to implementations that are often unreliable, not readily adaptable to changing requirements, exhibit long development cycles, and are over-dimensioned to satisfy performance targets. This leads us to pose the following question: *Why is it difficult to design such systems?*

A careful analysis of existing wireless embedded platform architectures, in conjunction with a survey of the state-of-the-art in the embedded systems design literature, reveals that the main problem is rooted in the *interference of hardware and software components in the time, power, and clock domains*.

One may think that straightforward sense-and-send applications do not suffer from this problem. However, in these scenarios several tasks must execute concurrently, e.g., reading measurements from sensors, processing measurement data, and transmitting packets. Sense-and-react applications such as cyber-physical systems [SLMR05] additionally feature control and actuation tasks. These concurrent tasks interfere when they compete for shared resources such as clock cycles, memory, and peripherals. While labor-intensive engineering may partially handle such resource interference in highly deterministic scenarios, this approach is not only unsustainable in the long term, but also ineffective if tasks are triggered by unpredictable events, e.g., in surveillance or tracking application scenarios [ABC+04]. Furthermore, as the system load increases, the effects of resource interference increasingly affect the timing behavior of individual tasks, which in turn adversely impacts overall system performance. Additional complexity is added to the system when incorporating power management techniques, which further exacerbates the problem.

Contributions. To address the above challenges, we advocate a novel approach to the construction of future wireless embedded platforms. We propose the functional separation of tasks onto a multi-processor architecture whereby the tasks interact through asynchronous message passing using a processor interconnect with predictable timing characteristics. Predictability of the interconnect entails that passing a message takes a known, bounded time irrespective of the attached processors. As a result, the proposed architecture decouples processors in the time, power, and clock domains, facilitating the *composable* construction of customized wireless embedded platforms. Composability not only gives the system designer the flexibility to select hardware and software components satisfying the needs of the application, but also ensures that their interconnection does not change the properties of the integrated parts [Jan06]. Furthermore, predictability of the message passing interface is essential to meeting the performance requirements imposed by certain application domains or wireless standards with tight latency constraints.

This chapter presents Bolt, the first processor interconnect that enables the composable construction of energy-efficient wireless embedded platforms. Bolt provides predictable asynchronous communication between two arbitrary processors, and thus decouples the processors in the time, power, and clock domains. Bolt sits between both processors and hides all complexities associated with handling asynchronous message transfers from system developers, while dissipating only negligible power. Two message queues, one for each direction, with first-in-first-out (FIFO) semantics form the basis of this processor interconnect. A signaling protocol facilitates concurrent message reads and writes on both queues, and indicates when there is at least one message ready to be read out from a queue and when a queue is empty. The two message queues and all internal state reside in non-volatile memory, thus preserving the state of Bolt independent of the power states of the two processors. Bolt requires only a minimal software interface with well-defined semantics and predictable timing to be implemented on each processor. This concept of a processor interconnect allows designers to choose arbitrary off-the-shelf or custom processors and "bolt" them together to create a customized dual-processor platform while avoiding resource interference.

This chapter makes the following contributions:

- We present the design of Bolt and detail a prototype implementation on a state-of-the-art low-power microcontroller.

- We apply formal methods to derive bounds on the worst-case execution time of Bolt's message passing operations.
- We detail Bolt's interface and how it should be used, before experimentally evaluating Bolt's performance with respect to power overhead, timing predictability and message throughput.

Using the presented Bolt prototype, we build a heterogeneous dual-processor platform consisting of a 32-bit ARM Cortex-M4 and a 16-bit TI system on chip. Extensive experiments with this platform demonstrate the following: *(i)* Bolt achieves power decoupling while incurring a negligible power overhead of 1.3 μW during periods of inactivity, *(ii)* the execution times of message passing operations exhibit empirical bounds that match the analytical bounds to within a few clock cycles, *(iii)* Bolt provides a high throughput of 1.5–3.3 Mbps for inter-processor messages that are 16–128 bytes in length, and *(iv)* Bolt enables the uninhibited concurrent execution of event-triggered sensing and wireless communication—a scenario that is complex, labor-intensive, and error-prone to implement on current wireless embedded platforms.

2.2 Background and Related Work

This section discusses the needs of low-power wireless embedded applications, the corresponding requirements on hardware and software components, and the problem of resource interference in state-of-the-art platform architectures.

2.2.1 Classical Mote Architectures

In the early days of wireless sensor networks, there was substantial interest in new hardware and software architectures for sensor nodes. These so-called *motes* typically provide an interface to sensors/actuators, process and communicate data.

A wide variety of wireless embedded platforms have been developed both in industry and academia that explore different points of the design space, such as Cricket [PCB00], Mica [HC02], Iris [Cro], EYES [vHDHK03], Telos [PSC05], Imote [NKA+05], BTnode [Beu06], TinyNode [DFFMM06], Camazotz [JSK+13], and OpenMote [VTWP15]. Attempts towards adaptability at run-time often utilize multiple resources of the same type having different properties, e.g., multi-radio platforms like BTnode [BDH+04], Opal [JKK+11], OpenMote+ [TPVW16], and Firestorm [AFC16], platforms with application-specific hardware acceleration like HaLoMote [EK16], and the concept of wake-up receivers [SBS02]. Both the platform architectures and the design principles used to realize them focused on finding the right trade-off between energy efficiency and the application-driven interfaces to sensors, computing and communication requirements, integration of state-of-the-art radios, small form factors, and autonomous operation [JHvdV+09].

2.2.2 Requirements of Modern Platform Architectures

The field of distributed low-power wireless embedded systems has matured to a point where now serious applications of societal and economic importance are within reach, such as the Internet of Things (IoT), industrial process control and supervision, environmental and structural monitoring, smart logistics, personalized medicine, home automation, and traffic control. In many of these applications, measurements are precious and must not be lost [XRC+04], so data must arrive reliably and in real-time [HJT12, RGR07], responses are safety-critical [DGA+05], and deployment and maintenance of a network is labor-intensive and costly. Thus, it is inevitable that distributed low-power wireless embedded systems become a high-quality infrastructure with known and predictable properties.

The requirements for successful hardware and software architectures in these domains are also much better understood than in the early days of Smart Dust [KKP99]. Developing software for heavily resource constrained hardware platforms is known to be highly labor-intensive due to the tight coupling between functional and non-functional properties on the one hand and detailed hardware properties on the other. Despite advances in model-based design techniques, specific models of computation, intensive distributed testing and verification cycles, distributed low-power wireless embedded systems are still error-prone. To make matters worse, the final installations are often deployed into hostile environments that are unknown at design time and exhibit dynamically changing properties.

The application domains differ substantially in their requirements and hence a single platform does not suite them all with respect to computation, communication, memory resources, available energy budget, and degree of integration [AC14]. Nevertheless, we can identify two common trends of modern wireless embedded platforms:

- *Increasing Resource Demand.* The program and data memory required to implement even seemingly straightforward functionality has grown significantly. Similar observations hold for the required computational resources that very often cannot be fulfilled by the 8-bit or 16-bit microcontrollers that were employed in the first generation of motes.

- *Need for Adaptability.* Application tasks such as sensing, actuation, computation, and communication are highly dependent on the occurrence of events, e.g., from sensor and radio interfaces, and the availability of energy. The same holds for the associated modes of operation such as radio states, microcontroller power modes, duty cycle, and clock frequency.

One of the early attempts to increase the amount of available resources was the Imote2 [NHS+08], at the cost of high sleep currents limiting its use in energy-constrained settings. More recently, there are new MCU generations appearing on the market that combine low sleep currents, short wake-up times, and rich peripherals with a relatively high compute power, such as the ARM Cortex-M family [ARM] used in products from Texas Instruments, NXP, ST Microelectronics, Silabs, Freescale, and Atmel. However, their availability only solves part of the problem inherent to classical mote architectures. Highly adaptive and event-triggered application tasks with widely varying resource requirements interfere on shared resources, and therefore violate the principles of modularity and separation of concerns.

To illustrate this resource interference problem, let us consider an application that requires high-rate sampling of sensors, for example, acoustic sensors in a structural monitoring application [XRC+04]. Due to the fact that the single processing resource, the microcontroller, also needs to handle time-critical events from the wireless communication component, the available computational resources may not be sufficient and interference between tasks is inevitable. As a second example, suppose an application requires a node to react quickly to events, for example, to quickly update an actuator or to localize signal sources depending on the pairwise differences in the arrival of events at different nodes. Again, limited resources and interference on the single processing resource lead to unpredictable behavior in such adaptive scenarios. This kind of resource interference can only be partially hidden by software abstraction layers, e.g., as provided by TinyOS [LMP+05] and Contiki [DGV04], leading to fragile, monolithic, and over-provisioned systems that are tedious to design, implement, debug, and maintain.

2.2.3 Modular Multi-processor Platform Architectures

In order to address some of the aforementioned deficiencies, modular platform architectures have been proposed that can be adapted to the needs of various application domains. Examples of such platforms include the 4-layer modular architecture in [PDCDLTR06], the MIT Media Lab modular platform [BP05], the stackable architecture [OBD+05], Epic [DC08], PowWow [BS10], and IHPstack mote [SGG12]. In all of these designs, components communicate via a shared bus or a set of shared standard interconnects such as UART, I2C, SPI, and 1-wire. Novel shared interconnects, such as the M-bus [KPK+14], have been specifically designed for ultra-low power connectivity.

The need for heterogeneous systems consisting of multiple components that dynamically match resources to the specific task at run-time, e.g., sleep modes, clock configuration, voltage scaling, and component deactivation, has been recognized by other communities as well, especially in the area of mobile communications. As a consequence, we increasingly see commercially available designs combining heterogeneous resources for ultra-low power applications, such as the LPC4300 [NXP] from NXP, and the VF3xxR and MKW2xDx [Fre] from Freescale. All these designs use bus-based interconnects or shared-memory communication between the components. One notable exception is the TI F28M3x series [Tex], which is specifically designed for application domains requiring safety certifications. Instead of relying exclusively on shared resources for communication, it contains a FIFO buffer for conflict-free communication at the cost of not being designed for ultra-low power operation.

It has been recognized that whenever multiple resources communicate, the use of shared busses and shared memory seriously hampers modularity [KdNA+14]. The major obstacle for application domains with high dependability and safety requirements, such as automotive and avionics, in adopting multi-core and multi-resource platforms is the inevitable interference on shared resources [KNP+13]. Thus, there is currently no accepted path to certification, which requires guarantees on correct timing and functionality [WGR+09].

The main consequences of using shared memory or shared busses for communication among components are the following:

- *Coupling of Power and Clock Domains.* To reach the goal of energy-proportional performance, where the energy consumed grows with the amount of useful task execution performed, systems use different power and clock domains. However, a shared bus couples these domains and requires tight coordination of power and clock management. As a result, the principles of separation of concerns and isolation of independent functionalities are violated, and power management requires the interaction of many hardware and software layers [AC14]. As a rule, unnecessary interference and dependencies introduced on the hardware layer can rarely be decoupled by means of higher-level software constructs.

- *Interference in the Time Domain.* Using shared memory, constructs like semaphores and locks allow for mutually exclusive access to shared resources. However, they also make the timing of activities on one resource dependent on that of activities on another resource, seriously violating composability and the possibility of independent design. In other words, the system must be redesigned for each newly introduced task. Similarly, the bus as a shared communication medium causes timing interference, leading to highly pessimistic timing bounds, e.g., when using protocols like first-come-first-serve, fixed priority, round-robin, or inefficient communication, e.g., when using partitioned protocols like TDMA or time-triggered architecture [KB03].

Due to these deficiencies, alternatives like distributed memory, asynchronous message passing, and queue-based communication are all well understood in their underlying concepts and widely used in distributed systems, multi-processor systems, and networks on chip [JT03, HGBH09]. However, these concepts have not yet been thoroughly investigated in the context of energy-efficient wireless embedded platforms. This chapter aims to fill this gap.

2.3 Design of Bolt

We introduce Bolt, a novel processor interconnect for energy-efficient wireless embedded systems. By providing bidirectional asynchronous message passing with predictable message transfer times, Bolt is a key building block for the communication between components on emerging dual-processor platforms to support energy-efficient applications with high processing demands and stringent timing constraints. Bolt simplifies the design of such applications by completely eliminating, or at least limiting, the interference among different hardware and software components on shared resources. As such, Bolt promotes a paradigm shift toward the composable construction of energy-efficient heterogeneous wireless embedded platforms.

2.3.1 Overview

As depicted in Figure 2.1, Bolt is a piece of integrated hardware and software that sits between two processors A and C. Bolt lets processors A and C asynchronously exchange messages while executing within their

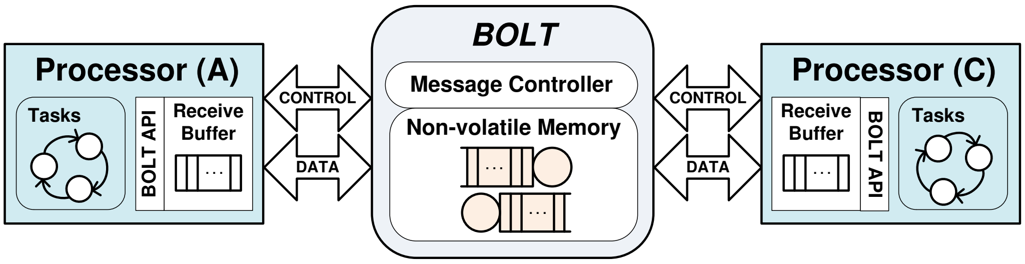


Figure 2.1: Overview of the Bolt processor interconnect. Bolt decouples processors A and C in time, power, and clock domains through predictable bidirectional asynchronous message passing.

own time, power, and clock domains. This ensures that each processor can independently write messages into Bolt or read messages out of Bolt. We design and implement Bolt in such a way that the execution time of read and write operations can be tightly bounded. Based on these bounds, we conceive a well-defined software interface by which each processor accesses Bolt in a non-blocking manner to exchange messages of variable length and possibly with different priorities.

The unique properties of Bolt allow for composable system designs. That is, a system designer can choose any two commercially available processors and existing software artifacts and integrate them to create a customized platform that satisfies the application needs, without changing the properties of the integrated parts. These parts can be separately designed, implemented, and validated, thus leading to modular systems that are easier to develop, understand, and maintain.

The key to achieving these advantageous properties is to tackle the problem of interference between different hardware and software components on shared resources. To solve this problem, we take a novel approach that is guided by the following established embedded systems design principles [HS07, Sif13]:

(1) Try to avoid resource interference.

(2) If resource interference is unavoidable, try to tightly bound it.

(3) Specify a formally verified interface with predictable properties.

We next describe how to apply the first and second design principles to the design of Bolt. Since the third design principle depends on a concrete implementation of Bolt as presented in Section 2.4, and a formal analysis of its timing properties as presented in Section 2.5, we discuss the interface by which the processors access Bolt in Section 2.6. While the basic concept can be extended to more than two processors, we focus on the dual-processor case for illustration purposes and its immediate applicability to the separation of concurrent application and communication tasks mapped onto processors A and C, respectively.

2.3.2 Platform Architecture

Bolt adopts asynchronous message passing to avoid interference between processors A and C wherever possible, as per design principle *(1)*. Thus, Bolt decouples processors A and C in the time, power, and clock domains.

- *Time.* Processors A and C interact with Bolt over dedicated control and data channels, as illustrated in Figure 2.1. Using these channels, messages can be asynchronously transferred between each processor and Bolt, irrespective of the state of the other processor. Due to asynchronous message transfer in combination with the buffering of messages until they are read out, Bolt effectively decouples A and C in time.

- *Power.* Bolt stores undelivered messages and all internal state in non-volatile memory. Thus, it also decouples the two processors in power, enabling independent power management of A and C over the maximum dynamic range, i.e., including deep sleep and power off modes. Bolt can also retain its own state and undelivered messages after a complete power failure, which may suddenly occur, for example, in energy harvesting scenarios.

- *Clock.* Finally, there exist implementation choices for the data channel that allow processors A and C to select and adjust their clock frequencies, decoupling them in the clock domain.

To achieve these properties, the control and data channel between Bolt and the attached processors must be serviced by independent hardware blocks, thus enabling simultaneous message requests and message transfers from either processor. To support independent bidirectional message transfers, Bolt stores undelivered messages in two FIFO queues, as illustrated in Figure 2.1. One queue is for messages written into Bolt by A and read out by C, while the other queue is for messages written by C and read by A.

Bolt's message controller manages the state of the message queues and the operation of the control and data channels. The message controller consists of software and hardware components. The latter can be realized on a multitude of hardware options, ranging from general-purpose microcontrollers to field-programmable gate arrays. A concrete message transfer is initiated asynchronously by one of the processors over the control channel, prior to the actual message transfer over the data channel.

Control Channel. Operating the control channel means coordinating data channel access for message transfers and indicating the availability of messages to the target processor. The signaling sequence of the control channel, as described in Section 2.4, ensures that each processor can initiate message transfers without causing inter-processor interference in the power and clock domains. Indeed, each processor is free to trigger a read or write operation at any moment in time over the control channel, and transfer the message over the data channel without interfering with the concurrent execution of the other processor.

For example, at the end of a write operation, Bolt uses the control channel to inform the target processor that there is now a pending message. It is then up to the discretion of the target processor when to inspect the control channel and when to initiate a read operation. Whenever the target processor is not busy performing local operations, it can read the message from Bolt using its dedicated control and data channels irrespective of the state of the other processor.

Furthermore, the power management of one processor is independent of the messaging operations invoked by the other processor. That is, one processor may, for example, choose to reside in a deep sleep mode for a given period of time as determined by its duty cycle, and is not explicitly woken up by the arrival of a message buffered in Bolt.

Data Channel. The actual message transfer between a processor and Bolt occurs over a bidirectional data bus that supports master/slave operation. Two popular examples of standardized busses supporting master/slave operation are SPI and I2C. Each interconnected processor is the master of its dedicated data channel. That is, each processor provides the clock required to transfer each bit over the bus, while Bolt always operates as a slave on the bus. This ensures that each processor can transfer messages using its own independent clock frequency, thereby decoupling both processors in the clock domain.

Bounding Unavoidable Interference. The architectural design decisions described thus far effectively avoid any interference between processors A and C in the power and clock domains. In the time domain, however, A and C may interfere. Since message transfers are asynchronous, it is possible that A and C simultaneously request a message operation on the same queue, e.g., when one processor wants to read a message, while the other processor wants to write a message. This situation creates unavoidable interference at the message controller, since the two processors compete for access to the same message queue.

Following design principle *(2)*, we intend to tightly bound the execution time of read and write operations despite the resource interference. To this end, the design and implementation of Bolt strives to reduce as much as possible the hardware and software complexity of the message controller. By consequently following this design guideline, we are able to accurately model the message controller and to determine tight bounds on the execution time of read and write operations using a model checker, as described in Section 2.5.

We acknowledge that if a message queue is full, a processor will not be granted access to write a message into Bolt, thereby blocking the write operation. We address this problem through the concept of virtual queues and appropriate buffer dimensioning, as discussed in Section 2.6.

2.4 Prototype Implementation

To demonstrate the viability of the Bolt design, we implement a Bolt prototype using the 16-bit TI MSP430FR5969 microcontroller running at an 8 MHz clock frequency. This particular microcontroller is well-suited due to its ultra-low power dissipation in sleep mode, the availability of built-in non-volatile storage in the form of ferro-electric random access memory (FRAM), an abundance of built-in peripherals, and its commercial availability at low cost. We next detail the hardware architecture of the MSP430FR5969, followed by a description of Bolt's software state machine that executes on the selected microcontroller.

2.4.1 Hardware Architecture

Figure 2.2 depicts a simplified hardware block diagram of the MSP430FR5969 including its built-in peripherals. The microcontroller is based on a Von Neumann architecture [JPC14], where the core accesses program and data memory using a shared bus. The core, FRAM, and all peripherals are attached to a shared 16-bit data bus and a 20-bit address bus. All program code, message data structures, and run-time variables are stored in the non-volatile FRAM of size 64KB. The instruction processing of the core may be interrupted by either the general-purpose input/output (GPIO) module or the direct memory access (DMA) controller using auto-vectored interrupt processing, whereby each peripheral interrupt is processed according to a fixed priority. The peripheral interrupt priority is defined in hardware such that the DMA controller has a higher interrupt priority than the GPIO module.

The GPIO module has two ports, namely PORT3 and PORT4, each of which supports several input/output (I/O) lines. An I/O line can be

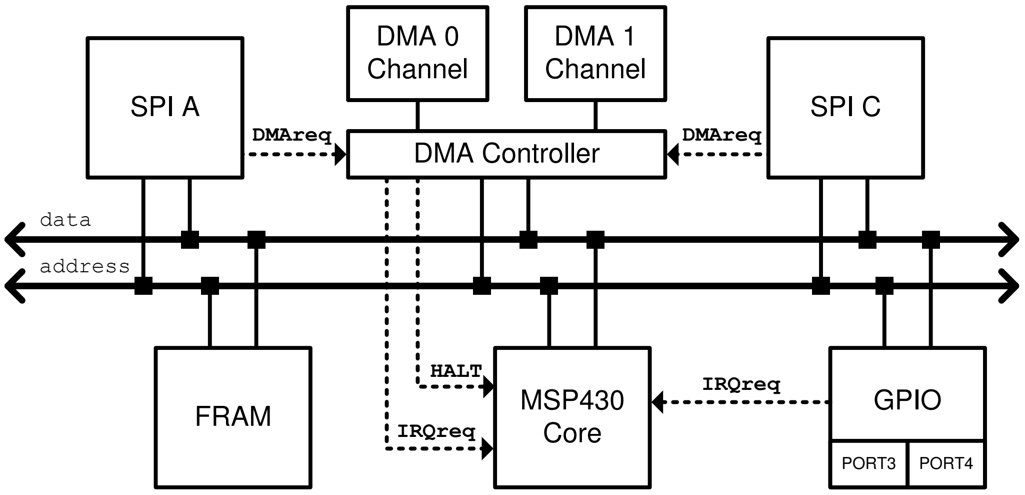


Figure 2.2: Simplified hardware block diagram of the microcontroller used in the prototype implementation of Bolt.

individually configured to initiate an interrupt on either a rising or a falling-edge. If an interrupt condition is detected on either port, the GPIO module will inform the core through an interrupt request *IRQreq*. In the case of simultaneous interrupt conditions, hardware prioritization ensures that the interrupt service routine (ISR) associated with PORT3 is executed before that of PORT4.

The microcontroller provides two independent SPI hardware modules, namely SPI A and SPI C. The byte-wise transfer between the SPI receive buffer and the FRAM is coordinated by dedicated DMA channels, that is, DMA0 for SPI A and DMA1 for SPI C. The DMA controller manages the bus arbitration and interrupt priority, with DMA0 having a higher priority than DMA1. When an SPI module has received or transmitted a byte, a hardware trigger is signaled to the DMA controller *DMAreq* to initiate a memory transfer, i.e., to store the received byte to FRAM or to fetch the next byte to transmit from FRAM. Before the DMA controller can perform such memory transfers, it must seize control of the shared address and data bus from the core. The DMA controller initiates this takeover by halting the core through a hardware-driven request *HALT*. The core is allowed to complete the current clock cycle before it is halted, thus relinquishing its control of the address and data bus. The byte transfer between SPI and FRAM takes precisely two clock cycles, after which the halt request is cleared by the DMA controller and the core resumes operation as bus master on the next clock cycle. Once a fixed number of bytes has been transferred, as determined by the DMA's software configuration, the DMA controller informs the core using an interrupt request.

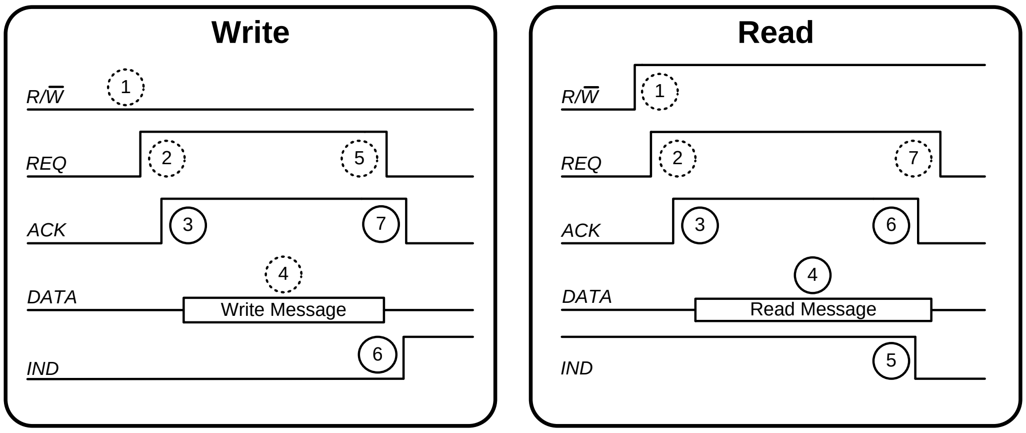


Figure 2.3: Signal sequences for write and read operations. The signals numbered with solid circles are driven by BOLT, while those numbered with dashed circles are driven by the connected processor.

Control Channel. A dedicated GPIO port is used to implement the control channel toward each interconnected processor. Specifically, PORT3 is assigned to processor A, while PORT4 is assigned to processor C. We adapt a four-phase level-sensitive handshake protocol [Cha84, Mye01] to provide coordinated access for reading or writing a message over the data channel, as illustrated in Figure 2.3. The control channel consists of the four lines $R/\overline{W}$, *REQ*, *ACK*, and *IND*. The $R/\overline{W}$ line defines the requested operation as either a message read or a message write. The *REQ* line is used by the interconnected processor to request the specified message operation, while the *ACK* line is used by BOLT to grant a message transfer over the corresponding data channel. If BOLT's message queues are empty or full, access to the data channel will not be granted for a read or a write operation, respectively. BOLT indicates to a processor when there is at least one message available to read using a dedicated *IND* line. The *IND* line is updated at the completion of every message operation.

Data Channel. A dedicated SPI module is used to transfer messages between BOLT and each interconnected processor. BOLT configures both SPI modules in slave mode, thus decoupling each interconnected processor with respect to their clock domains. Dedicated DMA channels facilitates fast message transfers between each SPI module and the message data structures stored in FRAM.

2.4.2 Software State Machine

Figure 2.4 illustrates BOLT's software state machine executing on the MSP430FR5969 microcontroller. Once the hardware is powered on, execution starts by initializing all state variables and message queues. If a queue contains undelivered messages, the *IND* line is asserted

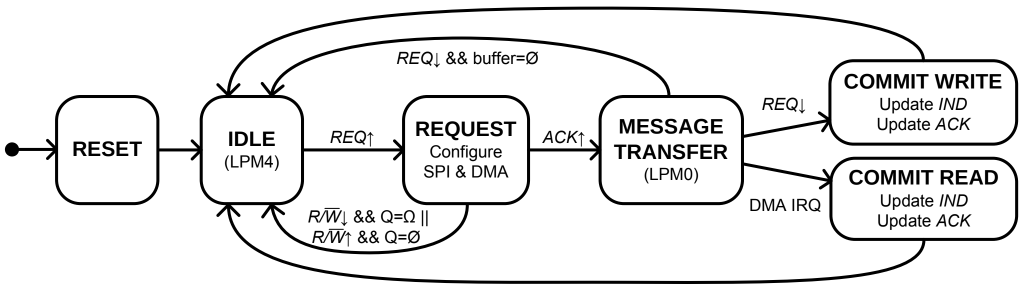


Figure 2.4: Bolt software state machine.

accordingly. The two *REQ* lines are configured as interrupt wake-up sources. The microcontroller is put into deep sleep (LPM4), while all other hardware blocks are turned off. The microcontroller remains in a deep sleep until a read or write operation is initiated.

When either processor initiates a read or write operation, i.e., by setting the $R/\overline{W}$ line and raising the *REQ* line, the associated IRQ wakes the microcontroller from deep sleep. Once the core is awake, the GPIO ISR will first determine the requested mode of operation and configure the corresponding SPI module and DMA channel for message transfer before updating the *ACK* line accordingly. The microcontroller will then enter a low-power mode (LPM0) with the core turned off, while the DMA and SPI modules remain active.

The completion of a message transfer is also processed by an ISR. However, depending on the message operation, either the GPIO or DMA ISR will be invoked. If the operation is a write, the falling edge of the *REQ* line signals the end of the message, thereby invoking the GPIO ISR. If the operation is a read, the DMA controller will trigger an interrupt toward the core. The ISR will update the internal data structures, update the *ACK* and *IND* lines, and disable the associated SPI and DMA peripherals. If there is an on-going message transfer involving the alternate processor, the microcontroller will return to the LPM0 sleep mode, otherwise it will return to deep sleep.

2.5 Formal Timing Analysis

As discussed in Section 2.3, two of Bolt's key design principles are to avoid resource interference wherever possible, and if interference is unavoidable, try to tightly bound it. Upper bounds on the interference on shared resources may be found using extensive measurements in a simulation environment or on a physical implementation of the system. However, this approach is extremely time-consuming and often infeasible in practice due to the number of possible combinations of input parameters and initial states of the system that need to be explored. An alternative approach, and one that has been well-studied in the embedded systems community, is to use formal methods such as model checking [BK08]. Through the construction of an accurate model of the system, one can apply rigorous mathematical tools to analytically derive safe bounds on interfering resources. We use such formal methods in verifying the timing predictability and functional correctness of Bolt.

In particular, we show next that Bolt has tightly bounded worst-case execution times for read and write operations, and that the physical interface operates according to its specification. We start by identifying the relevant hardware and software components of the Bolt prototype, and construct a model of their interactions using a network of timed automata [AD94]. We then parameterize the model based on a prototype implementation and formally verify the timing and functional properties of Bolt using the Uppaal model checker [LPY97].

2.5.1 Model Checking

The challenge in constructing a model of the Bolt prototype is capturing the complexity of several time-dependent and interacting state machines each of which has their own independent clock domain. In particular, processors A and C initiate message operations within their own clock domain, while the Bolt microcontroller responds to these operations using its own clock. Furthermore, contention between the execution of GPIO and DMA ISRs need to be formally modeled.

As a solution to this challenge we propose to model all interactions as timed automata and to extend each automaton with an independent clock variable. This technique is based on the theory developed in [AD94], where networks of time-dependent state machines interact through synchronization channels. This formalism allows us to model the complex interactions between Bolt and the interconnected processors by incrementing all clock variables synchronously.

We use Uppaal, a popular toolbox for the modeling, simulation, and verification of timed automata networks [LPY97]. We construct a run-time model of the Bolt prototype and its interactions with two interconnected processors using the following four interacting timed automata:

- The processor automaton
- The Bolt software state machine automaton
- The GPIO port automaton
- The DMA channel automaton

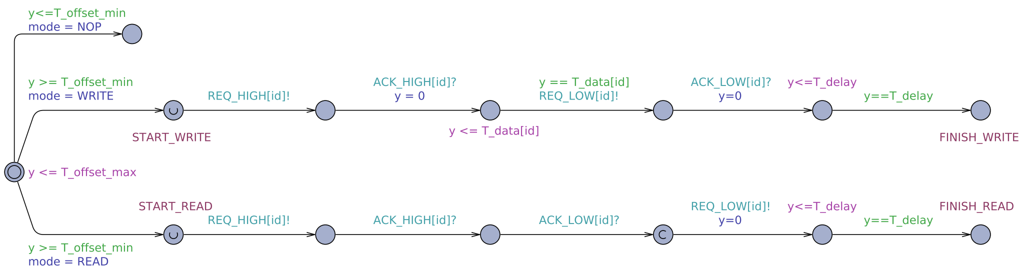


Figure 2.5: Platform automaton modeling the behavior of application and communication processors.

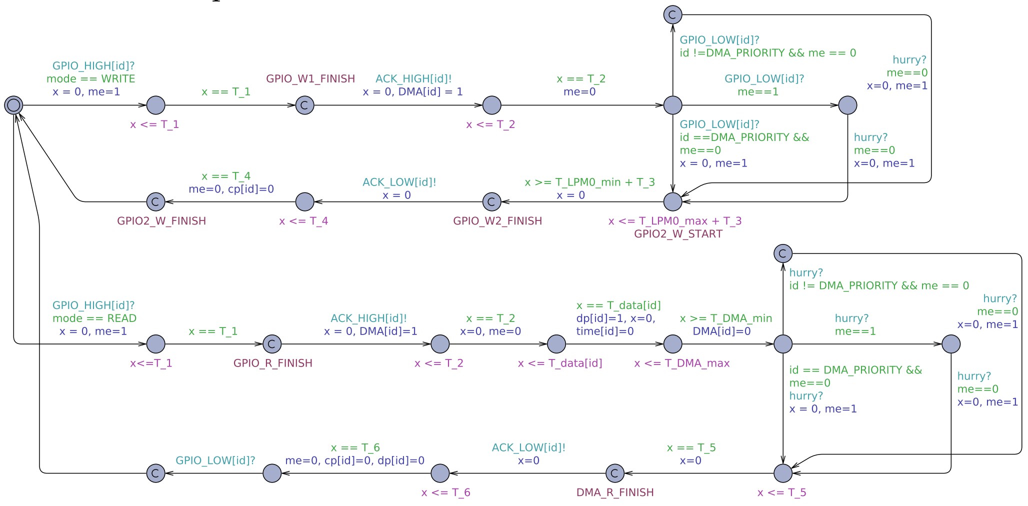


Figure 2.6: Timed automaton modeling the BOLT software state machine as represented in Figure 2.4.

We next introduce the behavior modeled by each automaton. Due to the inherent complexity of each automaton, we refrain from detailing each and every transition, but rather describe the overall functionality captured by the automaton.

Processor Automaton. The processor automaton, as illustrated in Figure 2.5, represents the expected signaling sequence of a read or write operation, according to the specification in Section 2.3. There are two instances of this automaton, one for each interconnected processor, and each with its own clock variable. The type of operation, i.e., read, write, or no-operation, and the time instant at which an operation is started are defined as run-time parameters.

BOLT Software State Machine Automaton. The BOLT software state machine automaton represents the software execution of the BOLT message controller. The automaton, as shown in Figure 2.6, uses state invariants and transition guards provided by UPPAAL [LPY97] to model the execution time of each GPIO and DMA ISR.

Non-deterministic transition times, such as the wake-up time from the

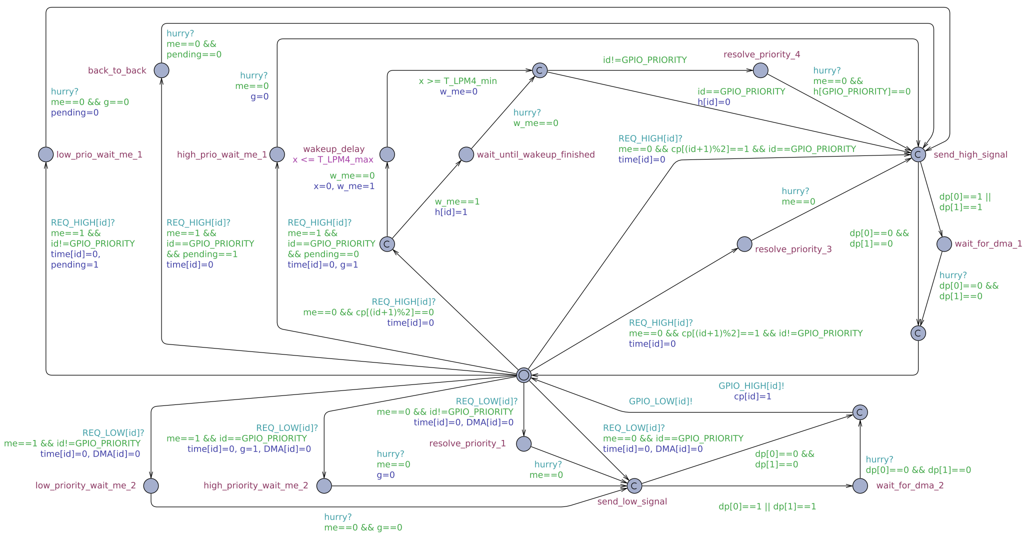


Figure 2.7: GPIO port automaton modeling the hardware prioritization between GPIO ports, between DMA channels and GPIO ports, and the non-deterministic wake-up delay from low power sleep mode.

LPM0 low-power mode of the core, are incorporated into the automata by limiting the invariant to at most the maximum delay, while relaxing the guard to at least the minimum delay. In this way, the Uppaal model checker is free to choose an arbitrary delay within the specified bounds and therefore explore all possible transitions. The execution of the GPIO and DMA ISRs are protected by a binary semaphore, which ensures that only one ISR can execute at any moment in time.

GPIO Port Automaton. The purpose of the GPIO port automaton is to model the hardware prioritization between the two GPIO ports. The final system model of Bolt contains two GPIO port automaton instances, one for each processor port. The GPIO port automaton, As illustrated in Figure 2.7, captures the detailed interaction between the two connected processors when performing simultaneous message requests. The automaton also takes into account that the GPIO ports have lower hardware priority than the DMA controller. That is, if a DMA interrupt is pending, the GPIO port automaton will ensure the DMA ISR acquires the semaphore first. The GPIO automaton also models the non-deterministic time to wake-up from LPM4 sleep mode. Global state variables are used to ensure that there is no wake-up delay when there are ongoing operations.

DMA Channel Automaton. During a read or write operation, the execution time of the GPIO ISR may be extended due to DMA memory transfers between the SPI peripheral bus and the FRAM. The DMA automaton, as shown in Figure 2.8, captures this complex interaction between ISR execution and hardware, while also ensuring the

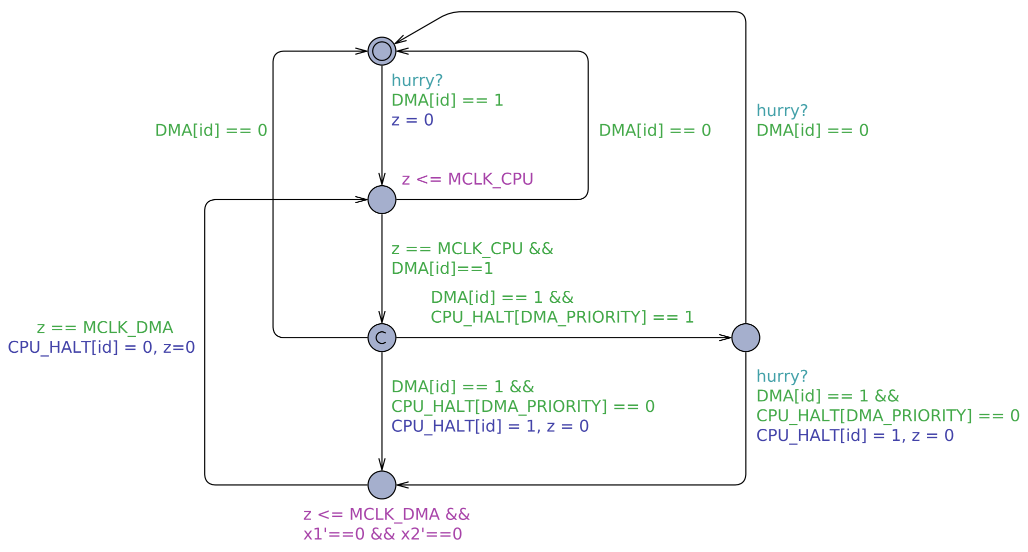


Figure 2.8: DMA channel automaton modeling the DMA hardware prioritization and the ISR execution time during DMA memory transfers.

Table 2.1: Summary of the timed automata in the Bolt system model.

Automaton	States	Transitions	Clocks
Processor	14	13	1
Bolt Software State Machine	24	28	1
GPIO Port	19	32	1
DMA Channel	5	9	1

prioritization between each DMA channel is adhered to. As described in Section 2.4.1, when there is a byte to transfer between the SPI and the FRAM, the DMA controller halts the core for two clock cycles. We model this using Uppaal's stop-watch feature [CL00], whereby a timed automaton can stop the clock of another automaton. This feature enables each DMA channel to stop the clock associated with the Bolt automaton for precisely two clock cycles before allowing it to resume.

Complete System Model. Table 2.1 lists the number of states, transitions, and clock variables of the individual timed automata that combine to precisely capture the behavior of the Bolt prototype. The complete system model comprises of nine automaton instances, specifically, two processor automata (for processors A and C), two Bolt software state machines, two GPIO port automata, two DMA channel automata, and one supplementary automaton that enforces the urgent selection of specific transitions in the network of timed automata. The resulting system model consists of 125 states, 165 transitions, 8 clock variables, and 15 synchronization channels.

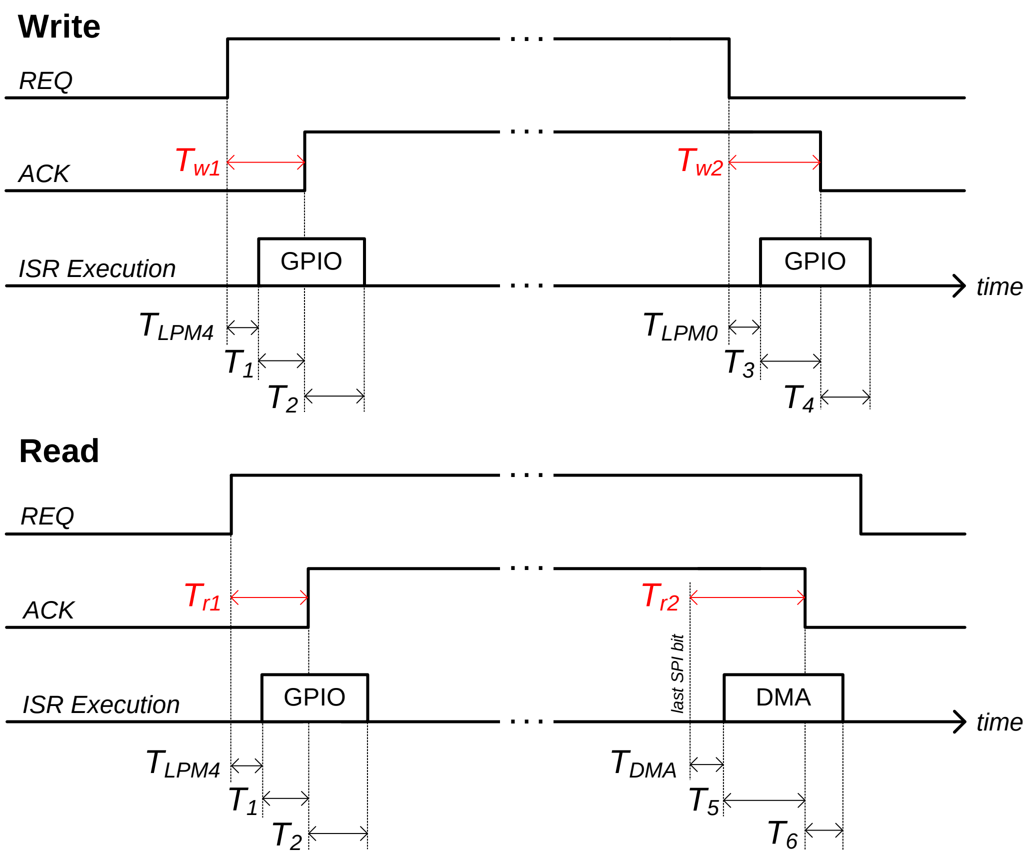


Figure 2.9: The execution of GPIO and DMA interrupt service routines during read and write operations.

2.5.2 System Model Parameterization

We next describe the methodology used to parameterize the Bolt timed automata system model using low-level measurements taken from the prototype implementation.

Figure 2.9 illustrates the execution of the GPIO and DMA ISRs with respect to the *REQ* and *ACK* lines for both read and write operations. To determine the specific worst-case delays in terms of clock cycles of the MSP430FR5969, we first need to investigate two important issues, specifically, *(i)* unbalanced ISR execution time, and *(ii)* non-deterministic hardware delays.

Since the GPIO and DMA ISRs must always determine which of the two processors is to be serviced, conditional statements are needed to select the desired control flow, resulting in the ISRs exhibiting port-specific execution times. By inserting the appropriate number of no-operation instructions into the ISR, we ensure that the GPIO and DMA ISRs execute a constant number of clock cycles, irrespective of which processor triggers the interrupt.

The next problem to address is the microcontroller's delay in waking up from the low-power sleep modes, denoted by T_{LPM4} and T_{LPM0}, and

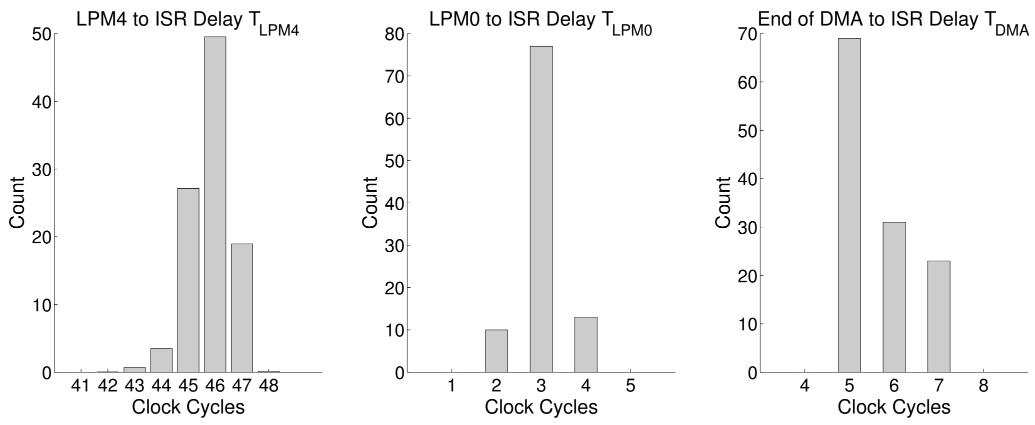


Figure 2.10: Histogram of microcontroller wake-up delays T_{LPM4}, T_{LPM0}, and DMA interrupt delay T_{DMA}, expressed in microcontroller clock cycles.

Table 2.2: Measured timing parameters of the Bolt system model in microcontroller clock cycles as annotated in Figure 2.9.

Parameter	**Description**
$T_1 = 172$	Start of GPIO ISR until rising edge of the *ACK* line
$T_2 = 48$	Rising edge of the *ACK* line until the end of the GPIO ISR
$T_3 = 149$	Start of the GPIO ISR until falling edge of the *ACK* line
$T_4 = 58$	Falling edge of the *ACK* line until the end of the GPIO ISR
$T_5 = 117$	Start of the DMA ISR until the falling edge of the *ACK* line
$T_6 = 59$	Falling edge of the *ACK* line until the end of the DMA ISR
$T_{LPM0} \in [2,4]$	Wake-up from LPM0 until beginning of ISR
$T_{LPM4} \in [41,48]$	Wake-up from LPM4 until beginning of ISR
$T_{DMA} \in [5, 7]$	Last bit on the SPI bus until the beginning of the ISR

the time it takes from signaling a DMA interrupt until the beginning of an interrupt context, denoted by T_{DMA}. Typically, these hardware-specific delays are specified in the datasheet of the microcontroller as being deterministic. However, detailed measurements on the Bolt prototype using a logic analyzer show that these particular delays are in fact non-deterministic, but are bounded within a small range. Figure 2.10 illustrates the histograms of the measured delays expressed in clock cycles of the MSP430FR5969 microcontroller.

With the measured upper and lower bounds of the timing delays, as listed and defined in Table 2.2, the formal system model of the Bolt prototype is fully parameterized. The timed automata model can now be used to determine the worst-case execution time of Bolt's complex run-time dynamics.

Table 2.3: Worst-case number of clock cycles for single read & write operations, and simultaneous read & write operations as derived from the UPPAAL model.

GPIO Port	Singular Write		Singular Read		Simultaneous Write & Read			
	T_{w1}	T_{w2}	T_{r1}	T_{r2}	T_{w1}	T_{w2}	T_{r1}	T_{r2}
PORT3	220	153	220	124	418	397	418	357
PORT4	220	153	220	124	466	397	466	357

2.5.3 Timing Predictability Analysis

In order to determine the worst-case execution time for read and write operations, that is, T_{r1}, T_{r2}, T_{w1} and T_{w2} as annotated in Figure 2.9 and defined in Table 2.2, we extend the UPPAAL model in three ways. First, we add an additional global clock variable to support the construction of verification queries using a common time base. Second, we modify the processor automaton such that the UPPAAL model checker can non-deterministically select the type of operation, i.e., read, write, or no-operation, performed by each processor instance. Third, we introduce a delay at the beginning of each operation and allow the model checker to non-deterministically select the delay of each operation. By selecting an appropriate upper bound for the delay, e.g., double the expected duration of a single operation, it is assured that all possible single and simultaneous operations, with all possible relative timing offsets, are explored by the UPPAAL model checker with clock cycle resolution.

We determine the worst-case execution times by submitting queries to the UPPAAL model checker. Specifically, we query the existence of a global time value that exceeds a given threshold X for a specific BOLT instance and state. We apply binary search to find the minimum threshold X that satisfies the query. Table 2.3 lists the worst-case execution times for single read and write as well as simultaneous read and write operations as derived from the UPPAAL model.

We now use these values to determine the worst-case execution time for any BOLT operation, irrespective of the type of operation performed. We define the worst-case execution time of the request phase $T_{request}$ and the commit phase T_{commit} in terms of the maximum execution time of read and write sequences as defined in Figure 2.9. The worst-case execution times for the request phase and the commit phase are defined by (2.1) and (2.2), respectively.

$$T_{request} = \max(T_{r1}, T_{w1}) \tag{2.1}$$

$$T_{commit} = \max(T_{r2}, T_{w2}) \tag{2.2}$$

Table 2.4: Worst-case execution times in clock cycles for request and commit phases, irrespective of the type of Bolt message operation performed.

GPIO Port	T_{request}	T_{commit}
PORT3	418	397
PORT4	466	397

The results for each GPIO port instance are summarized in Table 2.4. As expected, the request phase execution time differs between the two GPIO ports due to different GPIO port priorities. However, the execution time of the commit phase is identical for both GPIO ports, since the DMA controller halts the core for the same number of clock cycles under the worst-case scenario.

2.5.4 Functional Correctness

We define functional correctness of Bolt in terms of the asynchronous signaling sequence defined in Section 2.3. Given a valid input signal sequence, Bolt should always provide a correct output sequence. In terms of the developed timed automata system model, the formal validity of Bolt's functional correctness is assessed based on the existence or absence of a deadlock. Here, a deadlock denotes a state in which no edge transition can be executed by any automaton in the system model, that is, the system "stalls." After verifying the absence of such a deadlock in the developed model by submitting an appropriate query to the Uppaal model checker, we can conclude that the system model developed does not exhibit a state sequence leading to a deadlock.

2.6 Message Passing Interface

In real-world wireless sensing scenarios, application and communication processors must share information exhibiting differing priorities, while having to operate on intermittent energy sources, e.g., low-capacity batteries coupled with energy harvesting. We next detail how Bolt supports the prioritization of message flows and message consistency during power outages.

2.6.1 Message Structure and Prioritization

Bolt treats messages as a sequence of bytes, with the only constraint being the maximum number of bytes per message. It is therefore left to the system designer to construct an appropriate message structure

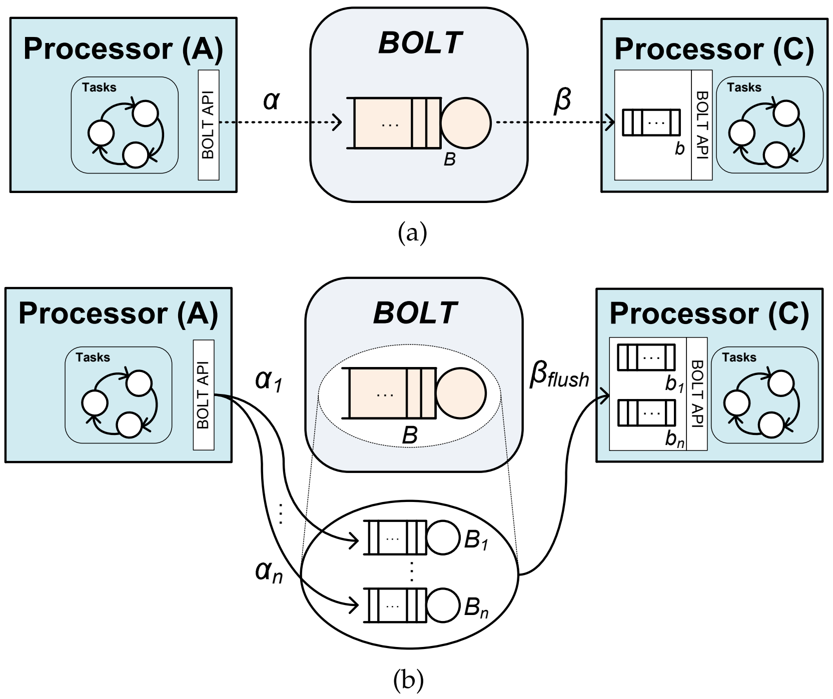


Figure 2.11: Dimensioning the length of Bolt's message queue to prevent blocking write operations for (a) single message prioritization, and (b) multiple message prioritization.

based on the needs of the target application. Experience has shown that incorporating a message type, the number of bytes stored in the message, and a cyclic redundancy check (CRC) are valuable message fields irrespective of the target application.

Due to the asynchronous message passing between processors, it is vital that Bolt does not block write message operations, i.e., prevent a write request due to a full message queue. Using the statistics on the rate of message writes α and the rate of message reads β, as illustrated in Figure 2.11(a), the length of Bolt's message queue B can be dimensioned using queuing theory [Kle76] to prevent blocking write operations.

Due to the complexity of real-world wireless sensing scenarios, messages types may be associated different priorities. For example, messages containing protocol data may be considered low priority, while messages containing protocol control information may be considered high priority. Since Bolt treats all messages transparently and stores them in a FIFO message queue, an attached processor has no way of determining if a high-priority message is waiting to be read out, and even if there is one pending, there is no mechanism to read the high-priority message first. Due to the aforementioned limitations, preventing blocking write operations with multiple message priorities becomes significantly more complex.

We therefore introduce an abstraction called virtual queues to reduce the problem of dimensioning multiple priority message flows to only dimensioning a single priority message flow. Without loss of generality, we assume messages from the application processor with priority i are written into a virtual queue of length B_i with rate α_i, and are read by the communication processor from a receive buffer of length b_i, as illustrated in Figure 2.11(b). We define the rate at which all messages in queue B_i are transferred into queue b_i as the flush rate. Under this construction, it is possible to compute the queue lengths B_i and b_i for each priority i under the assumption of a minimum flush rate β_{flush}. It follows that the length of the Bolt message queue B must be greater than or equal to the aggregate queue lengths of all prioritized queues, i.e., $B \geq \sum_{i=1}^{n} B_i$, while the receive buffer must be at least as large as all prioritized queues, i.e., $b_i \geq B_i$ for all message priorities i. The length of the Bolt message queue B to prevent blocking write operations can then be determined by network calculus [LBT01], e.g., using the real-time calculus toolbox [WT06], based on the statistics of the rate of message writes α_i for all priorities i, and the minimum flush rate β_{flush}.

The virtual queues abstraction makes it possible for a single processor to handle more than one input queue, e.g., in the case where Bolt is extended to more than two processors. If more than one thread having data dependencies executes on a single processor, asynchronous batch processing [CWA+10] may be applied to guarantee timing constraints. However, if more than one thread performing Bolt message operations executes on a single processor, a non-preemptive round-robin scheduler may be employed to ensure non-blocking message operations with predictable timing behavior.

2.6.2 Message Consistency

In real-world wireless sensing systems, the loss of power or the reset of a processor can lead to a loss of state. While the issue of reconstructing state or managing erroneous state, e.g., after a reboot, is left to application-specific high-level software, Bolt ensures that the messages it maintains are consistent. Specifically, Bolt's architecture and interface guarantees functional correctness of the processor interconnect in case of a spontaneous loss of power. This is achieved by storing both the Bolt message queues and all associated state in non-volatile memory, which will persist through a loss of power. Furthermore, the Bolt state machine ensures that partially read messages are not removed from the message queue, while partially written messages are removed from the message queue. This behavior ensures that undelivered messages can always be safely retrieved from Bolt when power is re-established.

Table 2.5: Bolt application programming interface.

Function	Description
`msg_t* read(int i)`	Read a message of priority *i* from the receive buffer
`void write(msg_t *m, int i)`	Write a message of priority *i* into Bolt
`int rtest(int i)`	Return the number of messages with priority *i* in the receive buffer
`int wtest(int i)`	Return the number of free messages in Bolt having priority *i*
`void flush(void)`	Read all messages from Bolt into the receive buffer

2.6.3 Application Programming Interface

Bolt provides a low-complexity application programming interface (API), as summarized in Table 2.5. Besides the `read` and `write` functions, it includes the three additional functions `rtest`, `wtest` and `flush`. When using Bolt, the following preconditions must be met by each interconnected processor:

- There must be space for at least one message in Bolt before `write` is called.
- There must be at least one message pending in the receive buffer before `read` is called.
- There must be no buffer overflow of the receive buffer.

The first precondition is satisfied by maintaining a local variable on each attached processor that stores the number of free messages in Bolt. This variable is initialized when the *IND* line is low, decremented by one at each `write` invocation, and accessed using the `wtest` function. The `wtest` function determines the number of free messages in Bolt by observing the *IND* line of the alternate processor, i.e., each processor must be able to access both *IND* lines. It follows that `wtest` must return a positive integer before a `write` is invoked. The second precondition is satisfied by storing a counter for the number of messages available in the receive buffer for each message priority *i*. A `read` therefore can only be invoked if `rtest` returns a positive integer for a specified priority level. The last precondition is satisfied by appropriately dimensioning the processor buffer according to the rate at which `flush` is invoked, as is determined by the application-specific flush rate β_{flush}.

If a message queue is empty upon a read request or the message queue is full upon a write request, the Bolt state machine will not raise the *ACK*

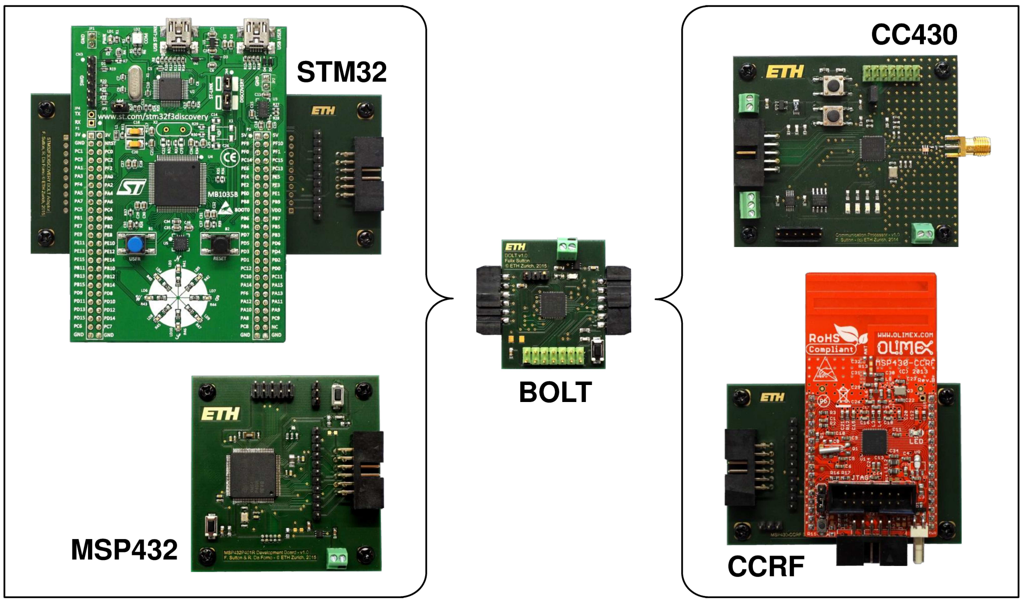


Figure 2.12: The Bolt prototype (middle) with two example application processors (left) and communication processors (right).

line in response to the raising of the *REQ* line. Instead, the Bolt state machine will safely return to an idle state and wait for the next request, while the requesting processor may stall indefinitely waiting for the *ACK* line. Nevertheless, the Bolt API ensures that an interconnected processor can perform a non-blocking read or write operation. This is done by monitoring the state of the *IND* line, maintaining local message counters, and appropriately executing the `wtest`, `rtest` and `flush` functions.

2.7 Experimental Evaluation

This section evaluates the performance of the Bolt prototype using extensive experiments on a custom-built wireless embedded platform that utilizes Bolt. We first consider a single Bolt instance and characterize its operation in terms of power decoupling and overhead, timing predictability, and message throughput. We then demonstrate the benefits of Bolt in the context of an event-triggered wireless sensing system.

2.7.1 Custom-built Dual-processor Platform

The prototype of Bolt is illustrated in Figure 2.12(middle) together with two examples of state-of-the-art application and communication processors that exhibit different computing capabilities and power dissipation, which have been prepared for interconnecting with Bolt.

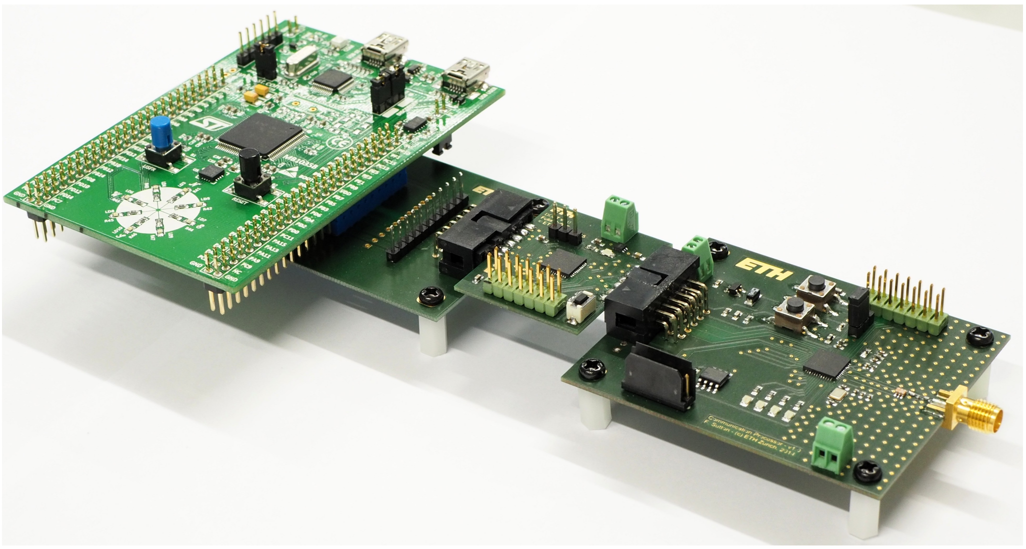


Figure 2.13: Custom-built heterogeneous dual-processor platform used in the experimental evaluation of Bolt. The Bolt prototype (middle) interconnects a 32-bit ARM Cortex-M4 (left) with a 16-bit TI CC430 SoC (right).

In order to experimentally evaluate Bolt, we construct a heterogeneous dual-processor platform, as illustrated in Figure 2.13, consisting of an STM ARM Cortex-M4 STM32F303VCT6 running at 72 MHz, and a 16-bit TI CC430F5137 running at 20 MHz interconnected by Bolt. We utilize this heterogeneous wireless embedded platform in the following experiments.

2.7.2 Power Decoupling and Overhead

We start by illustrating how Bolt decouples two processors in the power domain. To this end, we perform an example execution with the dual-processor platform, using an Agilent N6705A DC power analyzer to measure the current drain of the Bolt prototype, the Cortex-M4, and the CC430 at a supply voltage of 3.0 V with a sampling rate of 48 kHz.

Setup. Figure 2.14 illustrates the power dissipation of the three processors over a period of 4.5 ms, in which we can distinguish four distinct phases. In phase *(1)*, Bolt and the CC430 reside in a low-power sleep mode, while the Cortex-M4 collects sensor measurements by periodically sampling its built-in analog-to-digital converter (ADC). Bolt dissipates approximately 1.3 μW during this phase. In phase *(2)*, once enough samples are collected, the Cortex-M4 writes a single message into Bolt using an SPI frequency of 4 MHz. When the Cortex-M4 initiates the write operation, Bolt transitions from deep sleep into active mode, where it dissipates approximately 1.1 mW. At the end of the write operation, Bolt asserts the *IND* line to indicate to the CC430 that a message is pending and then returns to a low-power sleep mode. At the beginning of phase *(3)*, the CC430 wakes

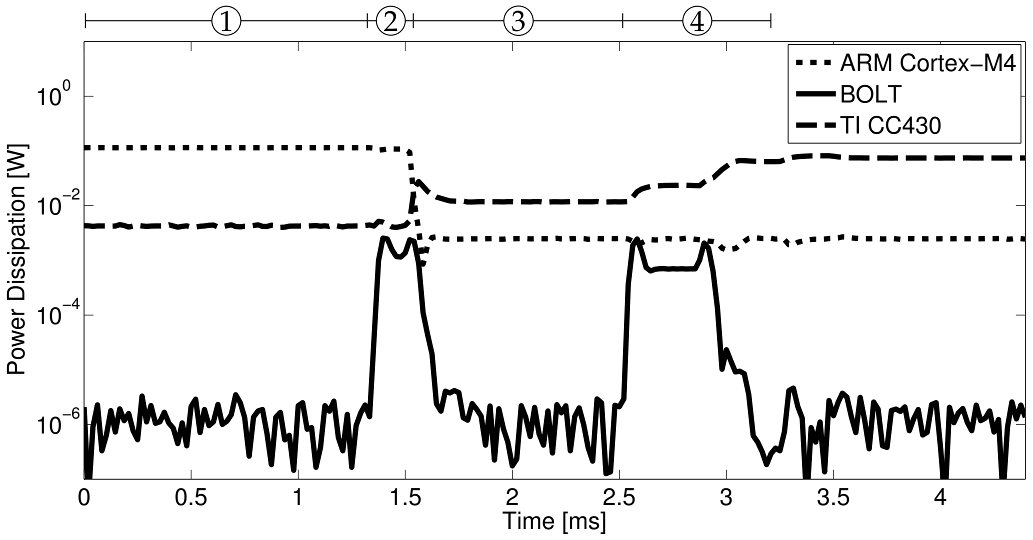


Figure 2.14: Power profile of the ARM Cortex-M4, TI CC430, and Bolt prototype.

up, initializes its built-in transceiver, and waits until the communication channel is available. During this time, the Cortex-M4 and Bolt both reside in energy-saving sleep modes. In phase *(4)*, the CC430 reads the pending message from Bolt using a 2 MHz SPI clock frequency and proceeds to transmit the contents of the message using its transceiver.

Results. The aforementioned experimental setup demonstrates that Bolt decouples two processors in the power domain. Indeed, each processor is free to locally decide when to enter or awake from a sleep mode, irrespective of the power state and utilization of the alternate processor and Bolt itself. This enables optimal power management over the maximum dynamic range supported by each attached processor.

When Bolt does not perform read or write operations, its power dissipation of 1.3 μW is less than state-of-the-art low-dropout voltage regulators, and in this particular setup, is several orders of magnitude lower than the sleep modes of both interconnected state-of-the-art processors, as visible in Figure 2.14. Furthermore, Bolt's active power dissipation of 1.1 mW is comparable to the sleep mode power dissipation of the Cortex-M4. We thus conclude that Bolt incurs a negligible power overhead, thus making it possible to realize energy-efficient heterogeneous wireless embedded platforms.

2.7.3 Timing Predictability

In a second experiment, we empirically verify the analytical bounds on the execution time of read and write operations determined by the Uppaal model checker in Section 2.5.

Setup. The time to complete a read or a write operation depends on the

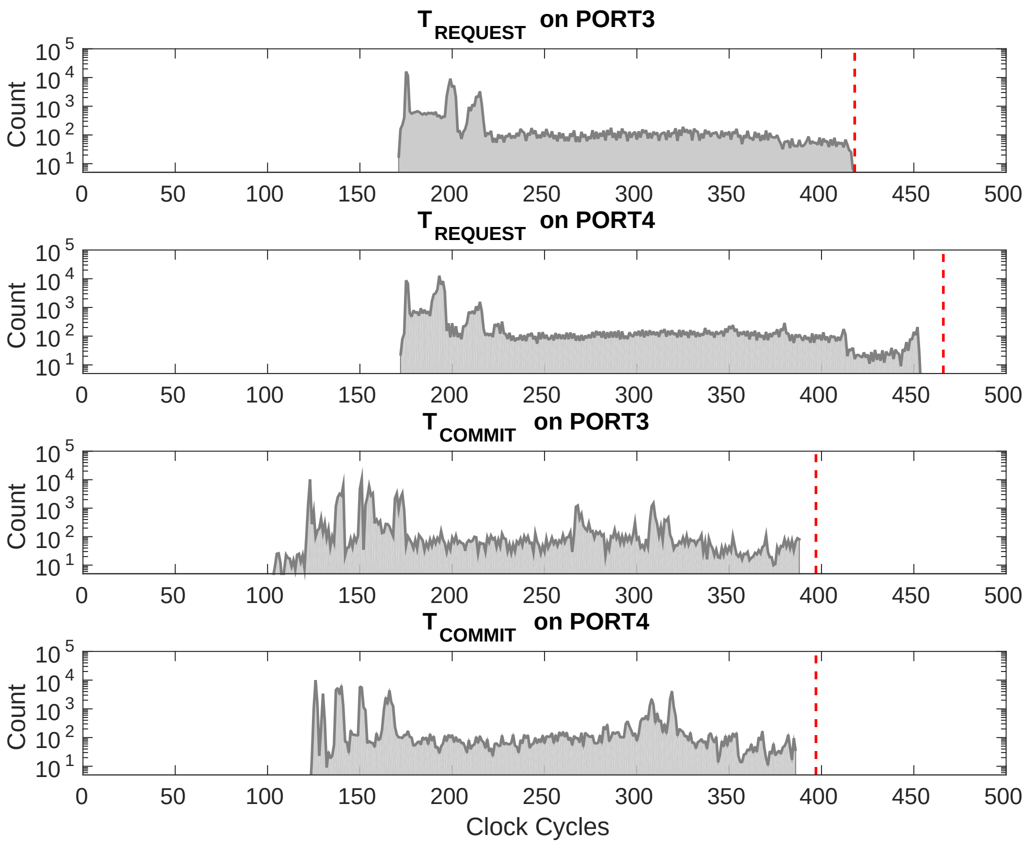


Figure 2.15: Histograms of the execution time of request and commit phases for concurrent message operations. Vertical dashed lines indicate the upper bounds determined by the UPPAAL model checker.

time to transfer the message over the SPI bus and the time BOLT needs to execute its interrupt-driven state machine according to the request and commit phases of the asynchronous interface, as detailed in Section 2.4. While the message transfer time is given by the message length and the SPI frequency, the duration of the request and commit phases, $T_{request}$ and T_{commit}, are non-deterministic due to interference on the same message queue in case of simultaneous accesses by both connected processors.

We accurately measure $T_{request}$ and T_{commit} on the BOLT prototype using a logic analyzer sampling at 25 MHz, while the Cortex-M4 and the CC430 perform concurrent message operations. We configure each processor to write a fixed number of messages into BOLT, before reading out all pending messages. The Cortex-M4 writes 48-byte messages and the CC430 writes 24-byte messages, resulting in equal message transfer times over the SPI busses. A total of 100,000 simultaneous message operations are performed, using a randomized waiting time between successive message operations to stress-test BOLT with different message request patterns.

Results. Figure 2.15 illustrates the histogram of $T_{request}$ and T_{commit} measured for the Cortex-M4 attached to PORT3 and the CC430 attached to PORT4. We observe that the analytical bounds are extremely tight. Specifically, the analytical bounds are at most 12 clock cycles greater than the worst-case execution times measured experimentally. This can be attributed to the quest for simplicity in the design and implementation of Bolt as well as the accurate modeling using a network of timed automata.

Furthermore, in accordance with the Uppaal system model, the worst-case execution time of the request phase, $T_{request}$, is slightly longer, i.e., 48 microcontroller clock cycles, on PORT4 compared to PORT3, which is attributed to the interrupt prioritization of PORT3 over PORT4. Instead, the worst-case execution time of the commit phase, T_{commit}, is approximately equal for both ports, since DMA memory transfers performed during the commit phase, as described in Section 2.5.3, halt the MCU for an equal number of clock cycles for each DMA channel.

In summary, the results demonstrate that the Bolt prototype exhibits timing-predictable read and write operations, satisfying a key requirement for enabling the composable construction of wireless embedded platforms.

2.7.4 Message Throughput

We next quantify the maximum message throughput supported by the Bolt prototype.

Setup. We determine the throughput by measuring the time to write a sequence of messages into Bolt at a SPI frequency of 4 MHz. Specifically, we let the Cortex-M4 write 1000 messages into Bolt, and measure the message transfer time with a logic analyzer. We consider write operations as they take longer than read operations, i.e., a message can be read 29 microcontroller clock cycles faster than it can be written.

Results. Figure 2.16 illustrates the maximum queue length, i.e., the number of messages that fit into Bolt's message queue, and the average throughput for different message lengths. We see that the throughput increases with the message length, supporting up to a maximum of 3.3 Mbps for 128-byte messages. Assuming the available memory in Bolt is evenly allocated to both FIFO message queues, the number of messages that can be stored into each queue is inversely proportional to the message length, allowing up to 1075 16-byte messages to be stored. Thus, by choosing a suitable message length, Bolt can support applications with high inter-processor communication demands. We therefore conclude that Bolt has no practical impact on the responsiveness of custom-built heterogeneous wireless embedded platforms.

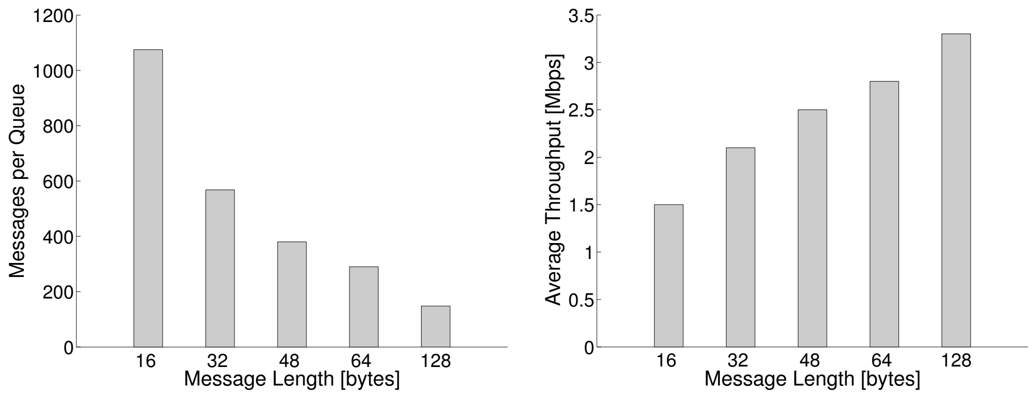
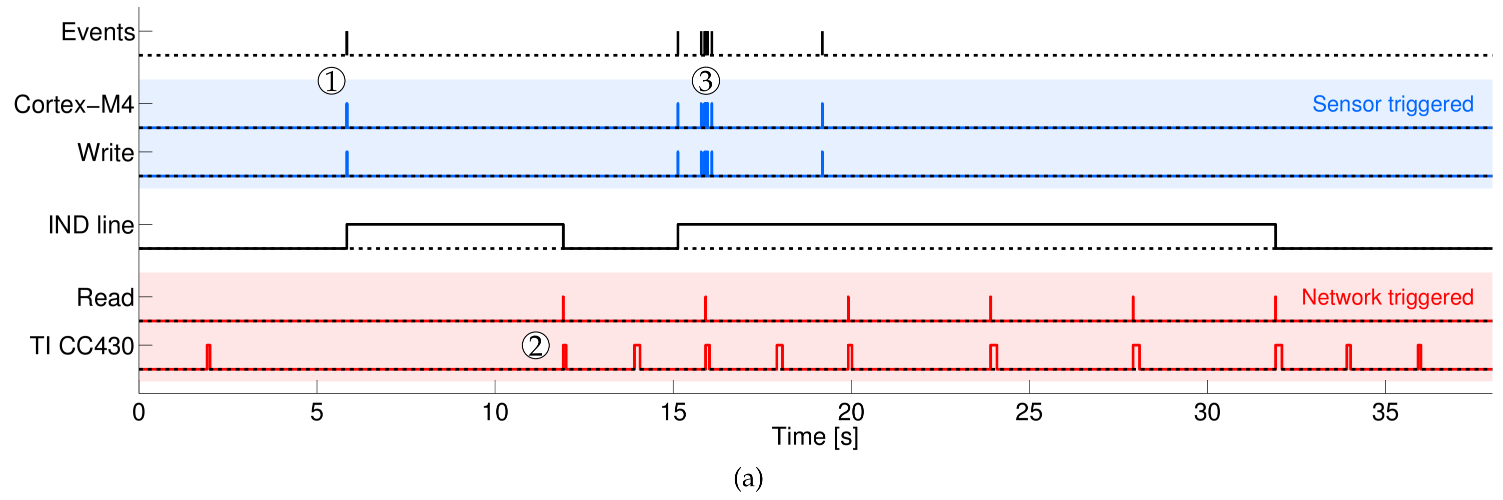


Figure 2.16: Maximum queue length (left) and average throughput (right) of the Bolt prototype for different message lengths.

2.7.5 Use Case: Event-triggered Wireless Sensing

In a final experiment, we demonstrate the benefits of Bolt in a typical event-triggered wireless sensing scenario. The application scenario demands resource-constrained wireless embedded devices to handle unpredictable events originating from a physical process under observation, while simultaneously servicing the wireless network interface to report those events to a remote host. Upon the detection of an event, a device starts to acquire ADC samples to classify the event, and constructs a message containing all data associated with the event. The time between event detection and the first ADC sample should be as short as possible so as to maximize the number of samples that contain useful data about the event. In order to report events and coordinate their operation, devices run a low-power wireless communication protocol. Given the timing constraints inherent to many protocols and standards, such as scheduled wake-ups [DDHC+10, EHD04] and communication slots [har, tsc], devices execute subject to hard real-time deadlines. During periods of inactivity, devices reside in a low-power sleep mode in order to maximize the operational lifetime of the system.

Realization without Bolt. Realizing this application based on a single-processor platform would lead to resource interferences whenever sensing and networking events must be handled at the same time, resulting in degraded performance or even erroneous behavior. By contrast, partitioning sensing and networking tasks onto application and communication processors, improves a device's capability to handle both types of events in a timely manner. However, using a shared memory or a shared bus for inter-processor communication would entail a tight coupling of the two processors in the time, power, and clock domains, with all the drawbacks discussed in Section 2.2.



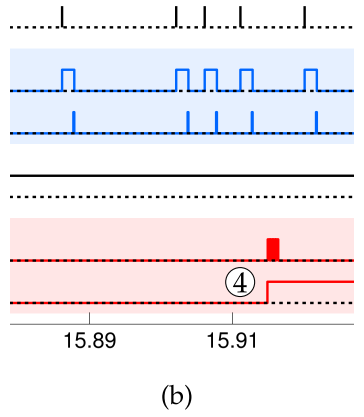


Figure 2.17: (a) Signal trace extracted from an example event-triggered wireless sensing application using Bolt, and (b) a magnified view of the signal trace at the time when the CC430 reads a sequence of messages from Bolt prior to the beginning of the LWB round.

Realization with Bolt. Using Bolt to interconnect the application and communication processors avoids these issues and provides maximum flexibility in selecting appropriate processors for each task, for example, the Cortex-M4 as a capable application processor and the CC430 as a ultra-low power communication processor. Furthermore, software implementation and re-use are simplified, processors can independently switch power modes while exchanging messages within guaranteed timing bounds, and the overall system becomes significantly more robust and easier to maintain.

The costs of Bolt relative to directly connecting two processors via I/O lines and a SPI bus is application-specific and difficult to quantify. Besides the additional cost of the microcontroller, there is an increase in the overall hardware surface area, i.e., the Bolt prototype requires 49 mm^2 plus support circuitry. The total power dissipation of the platform increases by 1.3 μW during periods of no message exchange. Each message transfer consumes approximately 350 nJ assuming 128-byte messages at a 4 MHz SPI frequency, while the message throughput reduces by about 17 % compared to an optimally configured SPI bus and under the best-case assumption that both processors are always ready to service the SPI to read and write messages. We believe the benefits of Bolt far outweigh these costs.

Setup. We implemented the above application on the custom-built dual-processor platform detailed in Section 2.7.1. The CC430 executes the Low-power Wireless Bus (LWB) [FZMT12], a communication protocol that uses fast and highly reliable Glossy floods [FZTS11] to transmit packets among nodes in a multi-hop network.

Results. A representative execution trace of the interaction between the application and communication processors through Bolt is illustrated in Figure 2.17(a). The top three rows show the arrival of sensor events, the servicing of sensor events, and the writing of event messages into Bolt by the Cortex-M4. The fourth row indicates the state of Bolt's *IND* line towards the CC430, and the last two rows indicate the reading of event messages from Bolt and the processing of network-triggered events by the CC430.

We observe from the first and last row that sensor events occur randomly and possibly in bursts, while network events occur periodically with the period changing according to the required communication bandwidth. By inspecting the active phases of both processors, we see that Bolt allows them to act independently of each other. For example, at time *(1)*, a sensor event triggers the Cortex-M4 to wake up from sleep mode and to write a new message into Bolt. The *IND* line indicates a pending message to the CC430. This message is then asynchronously requested from Bolt by the CC430 at time *(2)*, where it starts to interact with the network as determined by the communication schedule of the LWB [FZMT12]. The independent handling of sensor and network events ensures task deadlines are adhered to, even in cases of heavy event bursts as visible at time *(3)*.

As shown in Section 2.5.3, the time to perform message operations with Bolt can be tightly bounded. In this use case, this property enables the communication processor to read a sequence of pending messages just before they are sent over the radio via the LWB at time *(4)* in Figure 2.17(b), thereby limiting the number of wake-up cycles performed by the communication processor, and thus reducing the platform's energy consumption.

2.8 Summary

In this chapter, we focus on the resource interference problem of modern wireless embedded platforms. We argue that to address this problem, a multi-processor architecture built around an interconnect offering asynchronous message passing with guaranteed message transfer times is needed. Based on this idea, we design and implement Bolt, a processor interconnect for the composable construction of energy-efficient wireless embedded platforms. We show how the developed Bolt prototype avoids interference in the power and clock domains, while exhibiting tight bounds on unavoidable interference in the time domain between two interconnected processors. An experimental evaluation using a custom-built heterogeneous dual-processor platform demonstrates that Bolt can fully decouple the interconnected processors while providing a throughput of up to 3.3 Mbps, and timing-predictable inter-processor communication with a negligible power dissipation of 1.3 μW. We thus maintain that the architectural blueprint provided by Bolt is an important building block for the composable construction of next-generation energy-efficient wireless embedded platforms.

3

Design Methodology for Efficient Event-triggered Wireless Sensing Systems

The design and implementation of efficient event-triggered wireless sensing systems is difficult in practice due to the complexity of component interactions and the conflicting functional requirements of responsiveness, energy efficiency and adaptability. There is a need for a systematic design process that guides the system designer through the intricate component-level trade-offs which lead to an efficient implementation.

In this chapter, we present a novel system design methodology where well-established design principles, a logical model and a design guideline are used to transform the functional requirements of an event-triggered wireless sensing system into an efficient implementation. We first model the end-to-end sensing system as a pipeline of event-triggered components. We then model each component using a logical component model that encapsulates responsiveness, energy efficiency and adaptability design constraints. We then design and implement each component in isolation using a design guideline coupled with a range of design techniques. Finally, we integrate all components onto a platform architecture without sacrificing the underlying design constraints.

We exemplify our approach in the context of a real-world acoustic emission monitoring application. We design and implement acoustic emission detection, characterization and multi-hop dissemination components and integrate them onto a wireless embedded platform based on the Bolt processor interconnect. The experimental evaluation of the developed prototype demonstrates a wake-up time of 16 μs for the detection and 29 μs for the characterization of acoustic events, an average end-to-end latency as low as 113.2 ms for the multi-hop dissemination of acoustic events, and a total platform power dissipation of 59.7 μW during periods of inactivity.

3.1 Introduction

Wireless sensor networks have been widely used to monitor physical processes with high spatial coverage and temporal resolution, and with relatively low infrastructure costs [KW07]. In application domains such as structural monitoring [KPC+07], seismic monitoring [WAJR+05, EDR+11], and sniper localization [SML+04], the monitoring of *acoustic emissions* using a network of wireless sensors gives important insights into the state of the physical process under observation.

In order to capture the short-term and long-term dynamics of the underlying physical process, wireless sensing systems must support the following functional requirements:

- ***Responsiveness:*** The system must rapidly detect events, such as acoustic emissions, quickly characterize their specific features, and rapidly communicate the associated data through a multi-hop network to a remote host where analysis is performed.

- ***Energy Efficiency:*** In order to achieve long-term operation, the system must minimize the energy consumed by the detection, characterization and communication of events.

- ***Adaptability:*** The system must be able to adapt to changes in the environment. For example, it has to support the activation of additional sensor modalities, change the configuration of acoustic event characterization, or request on-demand communication bandwidth in response to a detected event.

Acoustic emissions are non-deterministic in nature, and as a result, wireless sensing systems for monitoring acoustic emissions are typically designed for always-on operation. This approach yields a system that is responsive and adaptable, but wastes significant energy resources during periods of inactivity. Instead, we aim to realize *event-triggered* wireless sensing systems that can simultaneously achieve responsiveness, energy-efficiency and adaptability.

Challenges. Transforming the aforementioned functional requirements into an efficient implementation is difficult in practice. The reason for this is twofold. Firstly, wireless sensing systems are a complex combination of analog, digital and software components, which makes it difficult for a system designer to comprehend all component-level dependencies and their unique design trade-offs. Secondly, the desired functional requirements of responsiveness, energy efficiency and adaptability are in conflict with each other, and thus component-level trade-offs are fundamental to achieving an efficient realization. For example, one may use an *always-on* approach where all components of the system are always active, resulting in a system that is responsive and adaptable, but poor in energy efficiency as there is significant energy waste during periods where there are no events of interest to be detected. Alternatively, a *duty-cycled/on-demand* approach may be used where components are periodically activated according to a duty-cycle or are only activated on demand. While this approach would drastically improve energy-efficiency, it will lead to *(i)* missed acoustic emissions since components are turned off at the instant the event arrives, or *(ii)* the delayed detection of acoustic emissions due to the time needed to activate components on-demand.

Approach. We simplify the system design problem by introducing a systematic design methodology that allows a system designer to quantify the component-level trade-offs between always-on and duty-cycled/on-demand schemes, thereby effectively managing the conflicting design goals. This is achieved by applying the well-established design principles [Kop08, Kop11] of partitioning, modularity, and abstraction, together well-known design techniques and a design guideline.

We propose to partition the wireless sensing system into a pipeline of logical event-triggered components. Each event-triggered component is characterized by design-time and run-time parameters, and adheres to an event-triggered interface specification. The design and implementation of each logical component proceeds in isolation using well-known design techniques coupled with the following design guideline: *sleep whenever possible, wake-up fast, and operate efficiently*. Components are then integrated onto a physical platform architecture, whilst preserving the underlying design constraints imposed on each individual component.

Contributions. This chapter makes the following contributions:

- We present a logical event-triggered component model for the construction of adaptable event-triggered wireless sensing systems, subject to responsiveness and energy efficiency design constraints.
- We exemplify our approach by designing and implementing a

wireless acoustic emission sensing system. We detail the design and implementation of an always-on ultra-low power acoustic sensor interface dissipating only 6.2 μW for the detection of acoustic emissions, an on-demand event-triggered characterization pipeline, and a duty-cycled event-based wireless protocol for the multi-hop dissemination of multiple acoustic emissions with an average latency as low as 113.2 ms.

- We demonstrate the integration of logical event-triggered components onto a novel dual-processor platform architecture. We experimentally evaluate its power dissipation for various run-time configurations, and show that the developed prototype exhibits a total power dissipation of 59.7 μW during periods of inactivity.

3.2 Event-triggered System Design

In this section, we present a design methodology for responsive, energy efficient and adaptable sensing systems that are suitable for certain classes of event-triggered wireless sensing applications. We next detail each step in the design process, before exemplifying its use through the realization of a wireless acoustic emission sensing system in Section 3.3.

(1) Partitioning. We begin by partitioning an event-triggered system into a pipeline of logical components. Each component represents a functional building block of the system and adheres to an event-triggered interface, whereby a component is triggered by an input event and produces an output event in order to trigger the next component in the pipeline. To give a concrete example, a wireless acoustic emission sensing system, which will be extensively detailed in Section 3.3, must first *(i)* detect an acoustic emission, then *(ii)* characterize the acoustic event, before *(iii)* disseminating the data associated with the event through a multi-hop wireless network. It follows that such a system may be partitioned into a directed graph of three logical event-triggered components representing the aforementioned functionality.

(2) Modeling. We then represent each logical component with an event-triggered model, as illustrated in Figure 3.1. The component model is specified by *(i)* its input and output interface consisting of an *event* and a *data* stream, *(ii)* its run-time adaptability represented by an *event filtering* random process, and *(iii)* its functional behavior represented by an *event processing* finite state machine. We next use an analytical framework to derive a components responsiveness and average power dissipation given the characteristics of its input event stream.

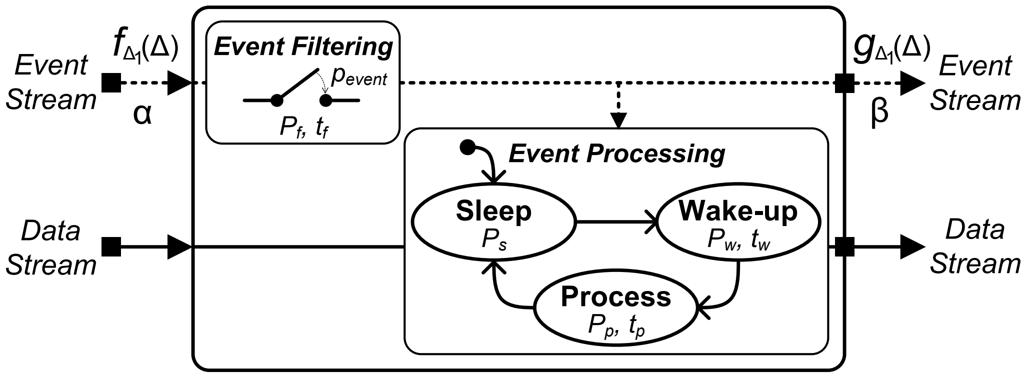


Figure 3.1: Model of a logical event-triggered component.

The *event filtering* element observes a stream of events at the input of the component, and depending on its run-time configuration, only a fraction of the events from the input stream will trigger the processing of the associated data stream. We represent this run-time adaption by the random sampling of the input event stream. In the context of acoustic emission sensing, an example of such event filtering is the configuration of the detection threshold. If the detection threshold is configured very low, all of the input acoustic events will likely be processed. However, if the detection threshold is configured very high, only a subset of input acoustic events will be processed. Another practical example is when classification techniques are used to only trigger the processing of a specific class of event. We model this run-time adaptability using a random sampling processes consisting of independent Bernoulli trials, where an event is processed with probability p_{event}, and is neglected with probability $1 - p_{event}$. The time and power dissipation of event filtering is parameterized by t_f and P_f, respectively. We assume that the energy consumption of event filtering is constant for all input events.

In order to determine the average power dissipation of the component, we first seek the characterization of the interarrival time between events after event filtering. We then use the parameterization of the finite state machine to determine the component's average power dissipation due to event filtering and processing.

As depicted in Figure 3.2, we represent the arrival time of an event i by t_i, where $t \in [0, \infty)$, and $i \in \mathbb{Z}$ represents the order of event arrival. We define the continuous random variable $\Delta_d^i = t_{i+d} - t_i$, to represent the time between events separated by arrival index $d \geq 1$. We note that when $d = 1$, the variable Δ_1^i represents the interarrival time of sequential events t_i and t_{i+1}, as shown in Figure 3.2.

Assuming event arrivals are independent and identically distributed, the interarrival times $\Delta_1^i, \Delta_1^{i+1}, \ldots, \Delta_1^{i+d-1}$ are independent continuous random variables characterized by the probability density function $f_{\Delta_1}(\Delta)$.

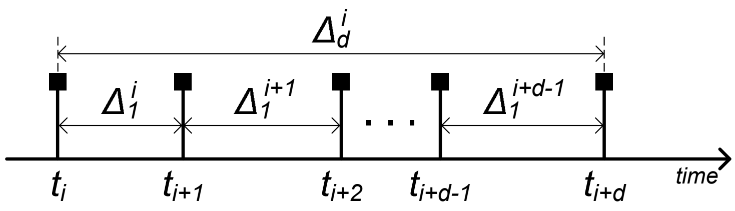


Figure 3.2: Example event stream at the input to an event-triggered component.

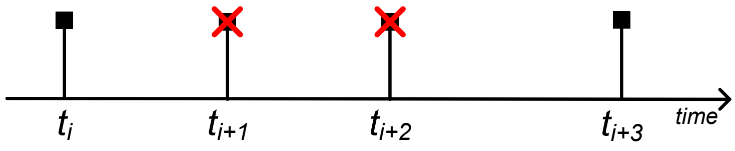


Figure 3.3: Example filtering of input events within an event-triggered component.

It follows that the average input event arrival rate α is determined from this characterization, as given by (3.1).

$$\alpha = \frac{1}{\int_0^\infty \Delta \cdot f_{\Delta_1}(\Delta)\, d\Delta} \tag{3.1}$$

The time between two events separated by an arrival index d, as denoted by Δ_d^i in Figure 3.2, is the sum of the continuous random variables that represent all the interarrival times between events at t_i and t_{i+d}, as given by $\Delta_d^i = \sum_{j=0}^{d-1} \Delta_1^{i+j}$. Using known properties of sums of continuous random variables [GS12], the probability density function of Δ_d^i is determined by the d-fold continuous convolution of $f_{\Delta_1}(\Delta)$, as given by (3.2). In other words, the probability density function of the time between d arrival indexes $f_{\Delta_d}(\Delta)$ is derived from the probability density of the interarrival times $f_{\Delta_1}(\Delta)$ using d-fold continuous convolution.

$$f_{\Delta_d}(\Delta) = (f_{\Delta_k} * f_{\Delta_{d-k}})(\Delta) \quad \text{for } 1 \le k \le d-1 \tag{3.2}$$

We now determine the probability of a filtered event having an arrival index of d to its predecessor. To give a concrete example, we consider the sequence of four input events as illustrated in Figure 3.3, whereby two events are neglected according to the aforementioned random sampling process. The probability of the last event having an index difference of $d = 3$ is given by the product of the probability of one event being processed and the probability of two events being neglected. In general, the probability p_d of a filtered event having an index difference d with its predecessor is given by (3.3).

$$p_d = p_{event} \cdot (1 - p_{event})^{d-1} \tag{3.3}$$

Using the expressions in (3.2) and (3.3), we now determine the probability density of interarrival times of randomly filtered events.

Specifically, the probability density function of the interarrival times after event filtering $g_{\Delta_1}(\Delta)$, is given by the probability of a filtered event having an index difference d multiplied by the corresponding probability density $f_{\Delta_d}(\Delta)$ for index difference d, summed over all possible index differences d, as given by (3.4). We note that due to the exponential decay of p_d as d increases, the summation may be approximated using a finite number of index differences.

$$g_{\Delta_1}(\Delta) = \sum_{d=1}^{\infty} p_d \cdot f_{\Delta_d}(\Delta) \tag{3.4}$$

It then follows that the average rate of events β produced at the output of a component is given by (3.5).

$$\beta = \frac{1}{\int_0^\infty \Delta \cdot g_{\Delta_1}(\Delta)\, d\Delta} \tag{3.5}$$

The *event processing* element specifies the functional behavior of a logical component with a finite state machine annotated with design-time parameters. A component resides in a default *sleep* state dissipating power P_s until it is triggered to execute. The arrival of an event after filtering will transition the component into a *wake-up* state, where the component awakes from sleep mode and performs necessary pre-processing tasks, taking time t_w and dissipating power P_w. The component then enters the *process* state, taking the input event and data stream and producing an appropriate output event and data stream, taking time t_p and dissipating power P_p. Once the component has completed processing, it immediately returns to the *sleep* state.

The responsiveness of a component is the time taken to filter an input event, process the event and produce an output event. The responsiveness is therefore given by the time to filter an event and the time spent in wake-up and processing states, i.e., $t_f + t_w + t_p$. We note that the component model does not assume event buffering, meaning that the responsiveness of a component must be less than the minimum interarrival time between sequential events, otherwise events will be missed.

The energy consumption per filtered event is determined by the sum of the energy consumed in all three states of the event processing finite state machine. Assuming constant and non-zero power dissipation within each state, the energy consumption u per filtered event is a linear function of interarrival time Δ and the parameterization of the finite state machine, as given by (3.6).

$$u(\Delta) = P_w t_w + P_p t_p + P_s\left(\Delta - t_w - t_p\right) \tag{3.6}$$

Since the interarrival time of filtered events Δ is characterized by the probability density function $g_{\Delta_1}(\Delta)$, we apply known results of functions of continuous random variables [Pap65] to determine the probability density function of $u(\Delta)$. Assuming the minimum event interarrival time is greater than or equal to the time to filter, wake-up and process the event, i.e., $\Delta \geq t_f + t_w + t_p$, then the probability density function of the energy consumption per filtered event $h_u(u)$ is given by (3.7).

$$h_u(u) = \frac{1}{P_s} \cdot g_{\Delta_1}\left(\frac{u - t_w(P_w - P_s) - t_p(P_p - P_s)}{P_s}\right) \tag{3.7}$$

It then follows that the average energy consumption of a component P_{avg} is the sum of the average power consumption of event filtering and event processing, as expressed in (3.8).

$$P_{avg} = \alpha P_f t_f + \beta \int_0^\infty u \cdot h_u(u)\, du \tag{3.8}$$

Therefore, given the statistical properties of the input event stream $f_{\Delta_1}(\Delta)$, the run-time adaptability parameter p_{event}, the energy consumption of filtering an event $P_f t_f$, and the design-time parameters of the finite state machine P_s, P_w, t_w P_p, and t_p, we can quantify *(i)* the properties of the output event stream using $g_{\Delta_1}(\Delta)$, *(ii)* a metric of component responsiveness given by $t_f + t_w + t_p$, and *(iii)* a measure of energy efficiency using the component's average power dissipation P_{avg}.

(3) Concretization. We now systematically design and implement each logical component in the pipeline according to the following guideline: *sleep whenever possible, wake-up fast, and operate efficiently*. This guideline not only resembles the overall responsiveness and energy efficiency design constraints, but translates directly onto the design-time parameterization of the logical event-triggered component model depicted in Figure 3.1. Specifically, a component must minimize the processing time t_p and enforce a low-power sleep state during periods of inactivity, minimize the wake-up time t_w in order to transition rapidly from sleep to the process state, and finally, minimize the power dissipation P_s, P_w and P_p in order to operate efficiently. Through the application of appropriate design tools, a concrete realization may be found which incorporates the components run-time adaptability, design-time parameterization, and additional domain-specific requirements.

(4) Integration. Now with a concrete realization of each logical component, we next integrate them onto a physical platform architecture whilst preserving adaptability, responsiveness and energy efficiency design constraints. In Section 3.3.4, we demonstrate this integration using a novel dual-processor platform architecture featuring *(i)* limited resource interference between components and *(ii)* support for composable construction using a formally verified interface between components.

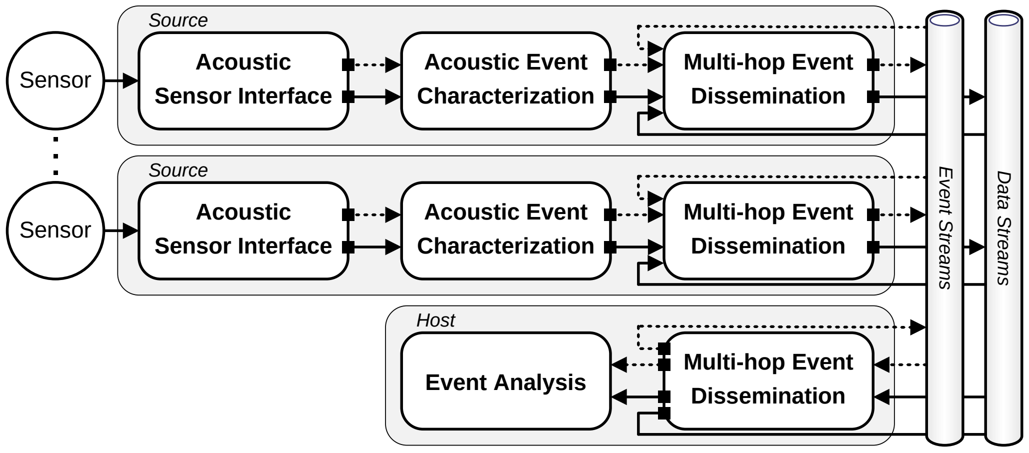

Source
Sensor
Acoustic Sensor Interface
Acoustic Event Characterization
Multi-hop Event Dissemination
Source
Sensor
Acoustic Sensor Interface
Acoustic Event Characterization
Multi-hop Event Dissemination
Host
Event Analysis
Multi-hop Event Dissemination
Event Streams
Data Streams

Figure 3.4: A wireless acoustic emission sensing system composed of logical event-triggered components.

3.3 Design and Implementation of a Wireless Acoustic Emission Sensing System

In this section, we demonstrate the design process introduced in Section 3.2 to construct a wireless acoustic emission sensing system. We consider a system where *source* nodes equipped with an acoustic sensor are deployed to detect, characterize and disseminate acoustic emissions to a remote *host* for analysis. As illustrated in Figure 3.4, we partition the system into a pipeline of logical event-triggered components, namely, *acoustic sensor interface*, *acoustic event characterization*, and *multi-hop event dissemination* components. We next detail the design and implementation of each logical component, and then demonstrate their integration onto a physical platform architecture.

3.3.1 Acoustic Sensor Interface

Related Work. Acoustic emissions are typically detected and characterized in commercial monitoring solutions such as [GO08], using a combination of an always-on powered analog circuits interfaced to a processor, as illustrated in Figure 3.5. The architecture consists of an acoustic sensor, i.e., a piezoelectric transducer, that converts dynamic motions into an electrical signal, followed by a fixed gain amplifier, bandpass filter and an analog-to-digital converter. This always-on acoustic sensing architecture has been widely adopted in the literature using a variety of different processors, including 16-bit microcontrollers [WAJR+05, KPC+07, CRM+08, SGFCPS+15, FPFP16], 32-bit microcontrollers [DPZMW+16, DSZ+16], DSPs [KBS08, TPM+15], ASPs [RGK12] and FPGAs [SML+04, VBN+07, LHV+09, DLGC15].

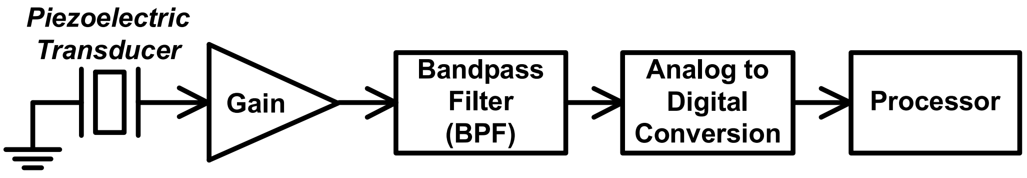


Figure 3.5: Block diagram of an always-on acoustic emission sensing system.

Table 3.1: Power dissipation of state-of-the-art acoustic sensor interfaces compared to the developed prototype.

System	Component	P_s
Lucid Dreaming [JKD+07]	Acoustic Sensor Interface (w/o LDO)	16.5 μW
CargoNet [MMF+07]	Acoustic Sensor Interface (w/o LDO)	3 μW
	Acoustic Sensor Interface (w/o LDO)	1.2 μW
This Work	Acoustic Event Characterization	2.5 μW
(with T = 15 s)	Multi-hop Event Dissemination	49.7 μW

However, since this architecture does not make use of a low-power *sleep* state, significant energy is consumed during periods of inactivity, i.e., when no acoustic emission of interest is observed. In order to reduce energy consumption, a *sensor-initiated wake-up* concept was proposed in [DGA+05], and further demonstrated in [JKD+07, MMF+07, LDX+10]. The key idea is to employ an ultra-low power analog circuit to wake-up the high power components only when an acoustic event is detected.

We extend this body of work by incorporating event-triggered wake-up *to all system components*, in conjunction with component-level run-time adaptability. As summarized in Table 3.1, the proposed approach achieves less than half the power dissipation P_s during sleep state compared to state-of-the-art acoustic sensor interfaces.

Application Requirements. The acoustic sensor interface component must satisfy the following requirements:

- ***Adaptable Detection.*** In order to support diverse operating conditions, the detection sensitivity of the interface must be configurable at run-time.
- ***Dynamic Range.*** It is important to not only detect very large signals i.e., having amplitudes of volts, but also detect very small signals, i.e., having amplitudes of millivolts.
- ***Noise Resilience.*** The sensor interface must be resilient against internal and external electrical noise sources, e.g., such as the

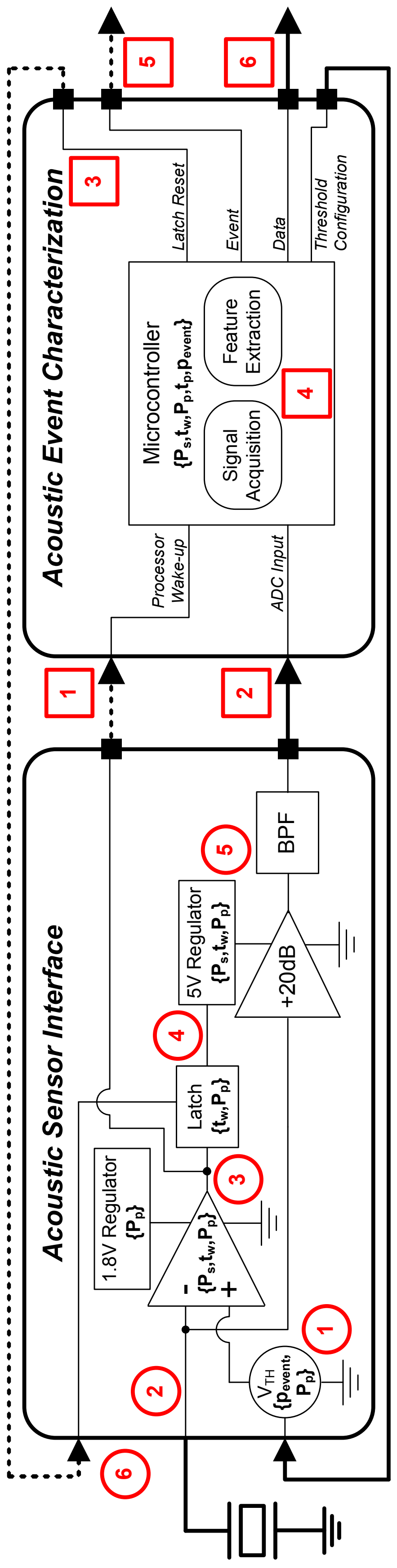


Figure 3.6: Block diagram of the acoustic sensor interface and acoustic event characterization components. The sequential operation of the acoustic sensor and characterization components is represented by the numbers inscribed in circles and squares, respectively.

noise associated with high-gain amplification, or an external power supply with a large ripple voltage.

Component Model. Figure 3.6 illustrates the block diagram of the acoustic sensor interface using the proposed event-triggered logical component model, and highlights the run-time and design-time parameters that drive the realization of the component. We next describe the behavior of the component, followed by its concretization through design space exploration, circuit simulation and rapid prototyping.

The operation of the acoustic sensor interface begins with *(1)* the configuration of the detection threshold V_{TH}. According to the application requirements this can be adjusted at run-time, and is therefore modeled by the random event filtering process with probability p_{event}. The generated threshold voltage provides a stable reference to an analog comparator *(2)*, which monitors the output of the acoustic sensor. When the sensor produces a voltage greater than the detection threshold, *(3)* the comparator generates an output event in order to trigger the next component in the pipeline, i.e., acoustic event characterization. The output of the comparator *(4)* also activates a latch, which turns on the voltage regulator supplying the amplification and filtering circuitry. The acoustic signal is then amplified and filtered according to application requirements, and *(5)* the output data stream is made available to the next component. Once the acoustic event characterization is complete, *(6)* an input event resets the latch, thereby turning off the high power amplifier circuit.

3.3.1.1 Design Space Exploration and Simulation

Threshold Generation. Since the threshold generation is always-on, we aim to minimize the active power dissipation P_p associated with the circuitry providing the programmable variable voltage source. Rather than employing a digitally controlled variable voltage regulator, we seek a solution that drains significantly less than 1 μA quiescent current. To this end, we employ a resistive voltage divider circuit [HH89] as defined by two series resistors, together with a bank of four parallel resistors each gated by n-channel transistors. Using SPICE circuit simulation, we select resistor values in the kilohm and megaohm range in order to support a threshold voltage range between tens of millivolts and several hundreds of millivolts, while as listed in Table 3.2, only dissipating a fraction of a microwatt.

Comparator and Voltage Regulator. We perform an extensive design space exploration to find the most suitable comparator and voltage regulator. The design space exploration considers the following five metrics, which directly relate to the parameterization of the logical event-triggered component model and the stipulated application requirements:

- *Quiescent Current:* We minimize the quiescent current of comparator and voltage regulator in order to decrease the power dissipation P_s during periods of inactivity.
- *Propagation Delay:* We minimize the comparator propagation delay in order to reduce the wake-up time t_w, and thus improve responsiveness.
- *Active Current:* We minimize the active current of comparator and voltage regulator in order to decrease the power dissipation P_p.
- *Minimum Input Voltage:* In order to maximize the sensitivity of the acoustic sensor interface, we seek a comparator with a very low minimum input voltage.
- *Power Supply Rejection Ratio (PSRR):* We maximize the PSRR of both comparator and voltage regulator in order to be resilient against external noise sources.

The design space exploration is performed in two distinct phases. The first phase reduces the search space by evaluating metrics of candidate devices using the respective datasheets, before selecting the three best performing devices. The second phase determines the most suitable device through the experimental evaluation of a rapid prototype.

Assessing the datasheets of 10 commercial comparators indicated a clear trade-off between the quiescent current and propagation delay metrics, where a lower quiescent current is achieved at the cost of a longer propagation delay. Due to the importance of these two metrics on the overall responsiveness and energy efficiency, we select the best three comparators, i.e., MAX920, MCP6546, and TS881, based only on a balance of these two metrics.

A custom printed circuit board is produced in order to experimentally evaluate the metrics of the three comparators under realistic conditions. Figure 3.7 depicts the experimental results by representing each metric on a unique axis, where the metric value has been transformed so that a higher value on the axis is more favorable. All metrics are evaluated using either a mixed signal oscilloscope or a precision multimeter, except for the PSRR, where the datasheet value is included for completeness. As represented in Figure 3.7, the MCP6546 comparator performs best compared to the other two devices, and is therefore chosen to implement the acoustic event detection component. The aforementioned design

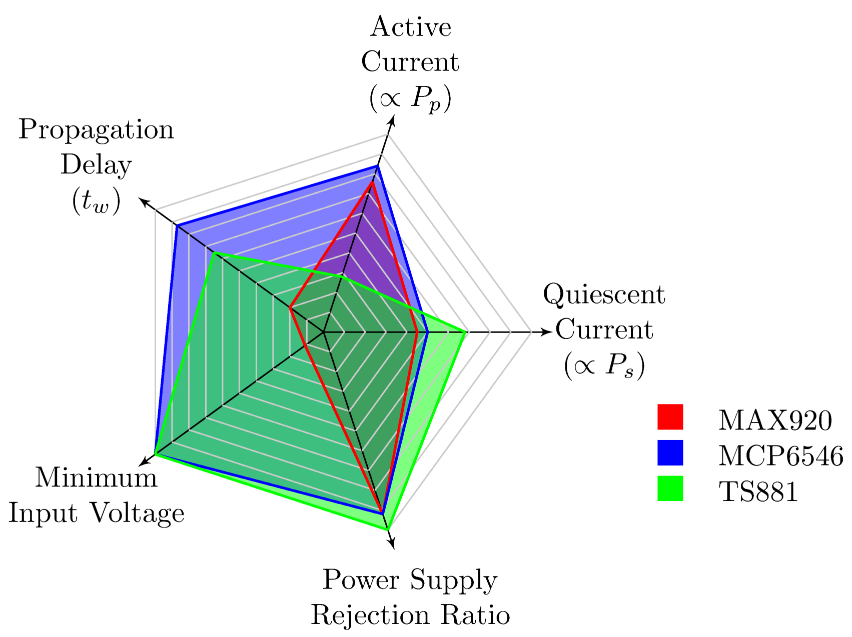


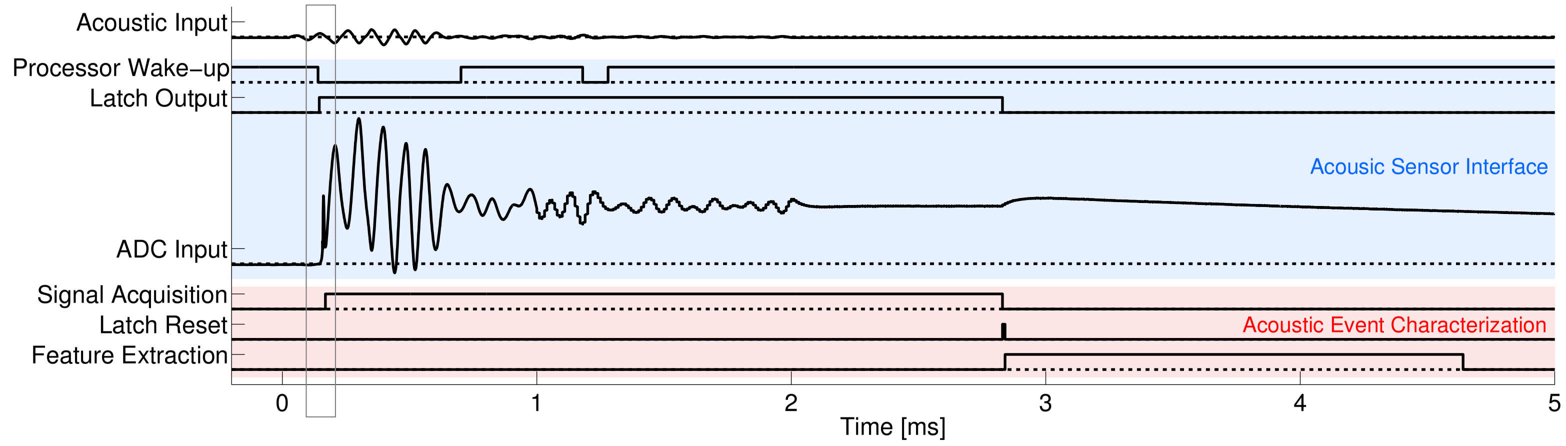
Figure 3.7: Visualization of the design space exploration for the MAX920, MCP6546, and TS881 comparators. The metrics are scaled so that a higher value on the axis is favorable, resulting in the selection of the MCP6546 comparator.

space exploration is repeated to find the most suitable voltage regulator, based on 13 commercial devices. In summary, the MCP1711 1.8 V regulator is selected due to its favorable active current, quiescent current, and PSRR metrics.

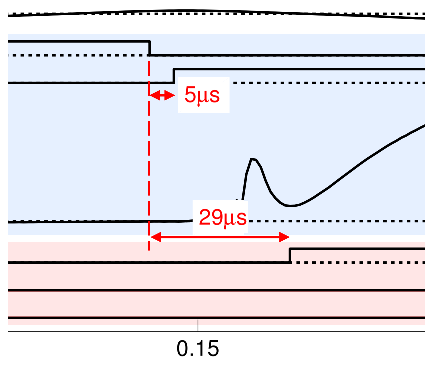
Latch. A digital latch provides a stable power gating of the amplification and filtering circuitry for the duration of an acoustic event. Instead of employing a commercial single-packaged latch designed for high-frequency operation and thus exhibits a quiescent current on the order of microamps, we design a custom set-reset (SR) latch [HH89] using n-channel and p-channel transistors. We verified the functionality of the latch using SPICE simulation, and measured its negligible power dissipation P_p of 45 nW and fast wake-up time t_w of 5 μs using a rapid prototype.

Amplification and Filtering. The voltage regulator for the high-gain amplifier circuit is chosen using a design space exploration, similar to that performed for the comparator. In addition to quiescent current and PSRR metrics, the *enable delay* metric was also considered in order to minimize the amplifier wake-up time t_w. The design space exploration considers 10 commercial devices, with the NCP603 5.0 V regulator chosen due to its low enable delay, and favorable quiescent current and PSRR.

The LM6482 operation amplifier is selected to implement the high-gain amplification, based on the frequency response of the acoustic sensor, and



(a)



(b)

Figure 3.8: (a) Signal trace of the acoustic sensor interface and acoustic event characterization components, and (b) a magnified view of the signal trace at the time when the acoustic event is detected. Measurements indicate that the wake-up time of the latch is 5 μs, while the wake-up time of the microcontroller implementing the acoustic event characterization component is 29 μs.

the application requirements placed on the amplification gain and output noise performance. A SPICE simulation is used to select the passive components determining the gain of the amplifier and the frequency response of the passive first-order bandpass filter.

3.3.1.2 Evaluation of the Acoustic Sensor Interface

Setup. We evaluate the responsiveness and power dissipation of the acoustic sensor interface through a controlled laboratory experiment. We emulated an acoustic sensor using an arbitrary waveform generator programmed with an acoustic signal extracted from a field deployment. We measured the analog and digital outputs using a mixed-signal oscilloscope, and measured the current drain using a precision multimeter.

Results. As illustrated in Figure 3.8(a), once the input acoustic signal surpasses the detection threshold, the comparator output triggers the wake-up of the acoustic event characterization component. After the 5 μs wake-up delay of the latch, the voltage regulator of the amplifier is turned on. This is visible in Figure 3.8(b) by the sudden spike on the analog-to-digital converter (ADC) input line. Once the ADC awakes from its sleep state, as indicated by the rising edge of the signal acquisition line in Figure 3.8(b), the acoustic signal is sampled according to application requirements. Once sufficient digital samples are collected, the latch is reset, and the acoustic signal is characterized.

The design-time parameterization of the acoustic sensor interface component is summarized in Table 3.2. We can conclude from these measurements that the acoustic sensor interface exhibits a power dissipation of 6.2 μW during periods of inactivity, supports a wake-up delay of 16 μs, and dissipates 5.6 mW during processing. The ultra-low power dissipation of the acoustic sensor interface supports multi-year operation using low-capacity coin cell batteries, and since acoustic signals are typically no longer than two milliseconds in duration, the rapid wake-up delay makes it possible to capture the predominant spectral components.

3.3.2 Acoustic Event Characterization

The purpose of the acoustic event characterization component is to extract important features from a detected acoustic emission in order to facilitate application-specific analysis at the remote host node. There are two common types of features for acoustic emission sensing, namely, *(i)* temporal features such as zero-crossing rate, rise-time, energy, and

Table 3.2: Parameterization of the acoustic sensor interface and acoustic event characterization components with respect to the time spent in wake-up and process states, and the power dissipation during sleep and process states.

Device	$\mathbf{P_s}$	$\mathbf{t_w}$	$\mathbf{P_p}$	$\mathbf{t_p}$
Threshold Generator	-	-	354 nW	Always-on
Comparator & Regulator	5.8 μW	11 μs	5.8 μW	2.5 ms
Latch	45 nW	5 μs	45 nW	2.5 ms
Amplifier & Regulator	22 nW	23 μs	5.6 mW	2.5 ms
Microcontroller	2.5 μW	29 μs	15 mW	4.3 ms

(ii) spectral features such as bandwidth, spectrum centroid, pitch.

While custom hardware blocks may be used to extract low-complexity features as exemplified in [GAJ+04, RGK12, KRG12, ACDB15, OBM+16], we instead utilize a state-of-the-art microcontroller to support complex and adaptive event characterization. We propose that the acoustic signal is first converted into the digital domain by an analog-to-digital converter (ADC), before computing a set of features according to the run-time configuration and application requirements.

Component Model. The block diagram of the acoustic event characterization component is illustrated in Figure 3.6, and is annotated with the critical design-time parameters used to facilitate its design and implementation. We next describe the behavior of the component followed by its concretization using a state-of-the-art low-power microcontroller.

The acoustic event characterization component is triggered by *(1)* an input event originating from the acoustic sensor interface. This event awakes the microcontroller from a deep sleep state, and *(2)* triggers it to begin sampling the acoustic signal using its built-in ADC. We use a built-in ADC in order to minimize the wake-up time t_w associated with peripheral initialization. Once sufficient samples have been collected, *(3)* an output event triggers the acoustic sensor interface to turn off its high-power amplifier. The microcontroller *(4)* computes the application-specific feature set, and produces an output event *(5)* indicating to the next component, i.e., multi-hop event dissemination, that characterization is complete and *(6)* a data stream is available. A second output stream is provided to configure the detection threshold of the acoustic sensor interface at run-time.

Since the event-triggered component model presented in Section 3.2 makes no assumptions on the buffering of streams between components, if the time to perform event characterization is longer than the time to digitize the input acoustic signal, then depending on the interarrival time between events, some events may never be characterized. We model this behavior using a random event filtering process with probability p_{event}, which provides an important design-time tool for appropriately constraining the wake-up t_w and process t_p times given the input event characteristics.

Concretization. While the choice of potential commercial micro-controllers is plentiful, we narrow the search space according to the parameterization of the logical event-triggered component model. Specifically, we seek *(i)* a microcontroller with a built-in ADC module supporting the application-specific sample frequency and resolution, *(ii)* a core that exhibits a low wake-up delay t_w from deep sleep, *(iii)* a low sleep power dissipation P_s, and *(iv)* a favorable active power dissipation P_p relative to the microcontroller clock frequency. After a survey of state-of-the-art microcontrollers, we select the MSP432 from Texas Instruments, as it combines the computational resources of a 32-bit ARM Cortex-M4 processor with the energy efficiency of the widely adopted MSP430 family of microcontrollers.

3.3.2.1 Evaluation of Acoustic Event Characterization

Results. We use the identical experimental setup described in Section 3.3.1.2 to measure the wake-up time t_w and power dissipation P_s and P_p of the acoustic event characterization component. Figure 3.8(a) depicts the timing behavior of signal acquisition consisting of 1000 14-bit samples at a sampling rate of approximately 400 kHz, followed by feature extraction consisting of the rise time, min/max amplitude, and signal energy. We note that using this specific implementation, the time taken for feature extraction is less than that for signal acquisition, and therefore we may assume $p_{event} = 1$ since sequential events will be successfully characterized, and $t_f = 0$ as event filtering imposes no additional delay on event characterization.

As illustrated in Figure 3.8(b), the time taken to wake-up the acoustic event characterization component is 29 μs. As summarized in Table 3.2, the measured power dissipation P_s during sleep state is 2.5 μW, and the power dissipation P_p during process state is 15 mW. Similar to the acoustic sensor interface component, the energy efficiency of the acoustic event characterization supports multi-year operation, while the responsiveness makes it possible to capture the dominant spectral components of a typical acoustic signal.

3.3.3 Multi-hop Event Dissemination

The purpose of the multi-hop event dissemination component is to rapidly and reliably deliver an event and its associated data to a host over a wireless multi-hop network. We next list the application requirements of this component, followed by a detailed analysis of how the proposed event-triggered logical component model is used to tailor a state-of-the-art wireless protocol for efficient multi-hop event dissemination.

Application Requirements. In order to analyze acoustic emissions at the host, the multi-hop event dissemination component must support the following properties:

- ***Time Synchronization.*** All source nodes must be tightly synchronized to a common time base, so to differentiate between acoustic emissions with respect to time.
- ***Reliability.*** The probability that transmitted acoustic event characteristics are successfully received at the host must be sufficiently high.
- ***On-demand Dissemination.*** A source node must be able to request network bandwidth on-demand for the dissemination of an event and its associated data.
- ***Simultaneous Events.*** The dissemination of simultaneous acoustic events must be supported, as the deployment of source nodes does not prohibit more than one source node from detecting the same acoustic emission.

Related Work. The past decade and a half of wireless sensor network research has produced a plethora of low-power wireless communication protocols, as surveyed in [HXS+13]. In this work, we concentrate on the *synchronous* class of protocols, as they support tight time synchronization by design, and have been shown experimentally to exhibit high end-to-end reliability. For example, Orchestra [DANLW15], a synchronous slot-based channel-hopping protocol for RPL and IPv6 networks, and Glossy [FZTS11], a synchronous flooding-based protocol, both support tight global *time synchronization* and demonstrate end-to-end *reliability* above 99.9% on testbed networks with up to 92 nodes.

While both protocols represent the state-of-the-art, we focus on Glossy, as its underlying time synchronization mechanism is achieved through radio-driven packet flooding using constructive interference, and therefore is insensitive to higher-layer protocol interactions such as routing. To this end, there have been several protocols proposed in the literature that build upon the Glossy flooding primitive, such as the Low-power Wireless Bus [FZMT12], Blink [ZMK$^+$17], Crystal [IMPR16], Pando [DLZL15], Splash [DCL13], and Chaos [LFZ13a], where each protocol has been tailored to a specific data dissemination scenario.

In this work, we choose to tailor the Low-power Wireless Bus (LWB), although we acknowledge that alternative protocols may indeed be feasible. We next present an overview of the LWB, followed by a detailed description of how we tailor the protocol to support the remaining application requirements of *on-demand dissemination* and *simultaneous events*.

3.3.3.1 Overview of the Low-power Wireless Bus

The open-source Low-power Wireless Bus (LWB) [FZMT12] transforms a physical multi-hop wireless topology into a logical shared bus using time-slotted Glossy floods. The LWB is structured using periodic communication rounds, having period T, where each round consists of a sequence of slots. Each slot is represented by a Glossy flood in which all nodes within the network participate. The radio of each node is turned off between rounds in order to save energy. The synchronization maintained by the Glossy flooding primitive ensures that each node awakes in time to participate in the next communication round.

We briefly describe the operation of the LWB using a single-hop network, as depicted in Figure 3.9. We note that while we only consider a single-hop topology, the operation of the LWB is equally applicable to multi-hop topologies. Every LWB round is initiated by the host, and begins with the *schedule* slot *(S)*. The schedule contains the structure and allocation of slots within the round. The next slot, called the *contention* slot *(C)*, gives the opportunity for a node to request a periodic data stream so to disseminate data to the host. The round finishes with a schedule slot, which informs all nodes of the next round period as computed by the host.

In the example illustrated, node S_1 detects an acoustic event that must be disseminated to the host for further analysis. During the next round, the node indicates its communication demands to the host, e.g., a periodic data stream specified by an inter-packet interval and a start time, by transmitting a stream request during the contention slot. Once the host receives the stream request, it will schedule appropriate bandwidth by allocating periodic *data* slots *(D)* to node S_1, and update the round period accordingly. During the next round, the schedule slot defines an *acknowledgement* slot *(ACK)* in response to the stream request, a data slot allocated to node S_1, together with the contention and schedule slots. Node S_1 then disseminates its event data to the host, possibly over several

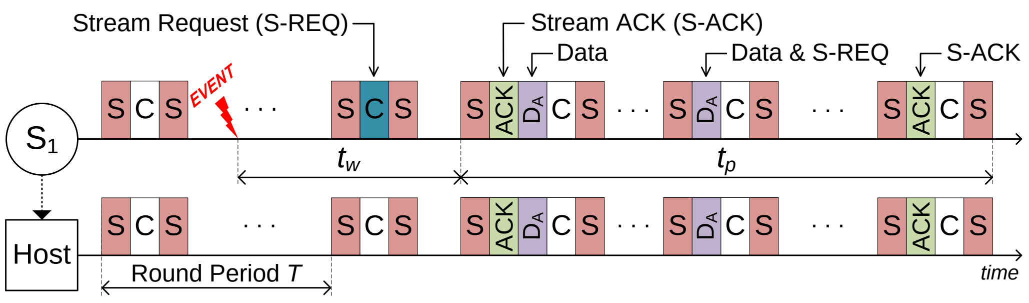


Figure 3.9: Example operation of the Low-power Wireless Bus (LWB), where source node S_1 detects an event and disseminates the associated data to the host by the setup and tear down of a periodic data stream.

rounds depending on the bandwidth allocated to it by the host. Once node S_1 has no further data to transmit, it piggybacks a stream request into its last data slot so to remove the data stream. The removal of the data stream is confirmed by the host in the next round using an *ACK* slot.

Component Model. The time-triggered operation of the LWB, i.e., sleep, wake-up, communicate, and return to sleep, is analogous to the event processing state machine embedded within the logical event-triggered component model introduced in Section 3.2. Specifically, we denote the time from an event arrival until the allocation of data slots as the wake-up time t_w, the time from the allocation of data slots until the removal of the data stream as the processing time t_p, and the average sleep state power dissipation P_s as the power dissipated for schedule and contention slots during round period T. We next describe the limitations of the LWB in supporting on-demand dissemination under the presence of contention, before detailing how we tailor the protocol in the context of the logical component model.

3.3.3.2 Event-based Low-power Wireless Bus (eLWB)

The LWB is designed to support data streams, making it particularly well suited to periodic data delivery with slowly changing traffic demands. However, when we consider event-triggered wireless sensing applications, the means by which periodic data streams are requested from the host give rise to the following challenges:

- Multiple source nodes may simultaneously detect an event, resulting in these nodes transmitting their stream request during the same contention slot. Due to the poor scaling of the capture effect, as experimentally evaluated in [LFZ13a], the probability of a stream request being successfully decoded at the host reduces significantly as the number of contending nodes increases, thus increasing the wake-up time t_w due to random back-off mechanisms.

- The LWB supports the *sequential* allocation of *periodic and fixed* bandwidth to source nodes, which is in contrast to the requirements of event-triggered wireless sensing, where the *simultaneous* allocation of *aperiodic and variable* bandwidth to source nodes is needed. Therefore, even if the challenge of contending stream requests is overcome, it will still take several rounds for all contending nodes to have their bandwidth demands administered by the host, thus adversely increasing processing time t_p.

We address these challenges by modifying the behavior of the LWB, which we term the *Event-based Low-power Wireless Bus* (eLWB), without increasing the average sleep state power dissipation P_s. Specifically, we *(i)* reduce the wake-up time t_w by notifying the host when *at least one* event has been detected using an event contention slot, and *(ii)* reduce the processing time t_p by providing fixed bandwidth for the event streams and on-demand bandwidth for data streams.

Event Contention Slot. According to the simulation results presented in [WLS14], the packet reception rate of concurrent transmissions with identical packet payloads is significantly higher than the concurrent transmission of packets with independent payloads. We use this result to improve the likelihood that the host is notified of at least one event in the case when multiple source nodes detect an event within the same round period. We achieve this by replacing the original contention slot with an *event contention* slot *(E)*, as illustrated in Figure 3.10, whereby each node wishing to disseminate an event transmits a packet with an identical payload, e.g., a packet containing the single byte 0x00. Additionally, the host disregards the validation of the cyclic redundancy check when processing the contention slot. All source nodes are informed of a successful event contention by inspecting the change of round period from T to T_{event}, as specified in the schedule slot immediately following the event contention slot.

Event and Data Rounds. Once the host is notified of at least one event within the network, an *event round* provides all source nodes an opportunity to *(i)* disseminate an event and *(ii)* request bandwidth for data stream dissemination. The event round consists of a schedule slot and a unique data slot for each node within the network. The data slot has a fixed length of 1 byte, which is sufficient to indicate an event type and the required bandwidth for the data stream. In the multi-hop example depicted in Figure 3.10, all source nodes are provisioned with a data slot during the event round, but only nodes S_1 and S_3 disseminate an event

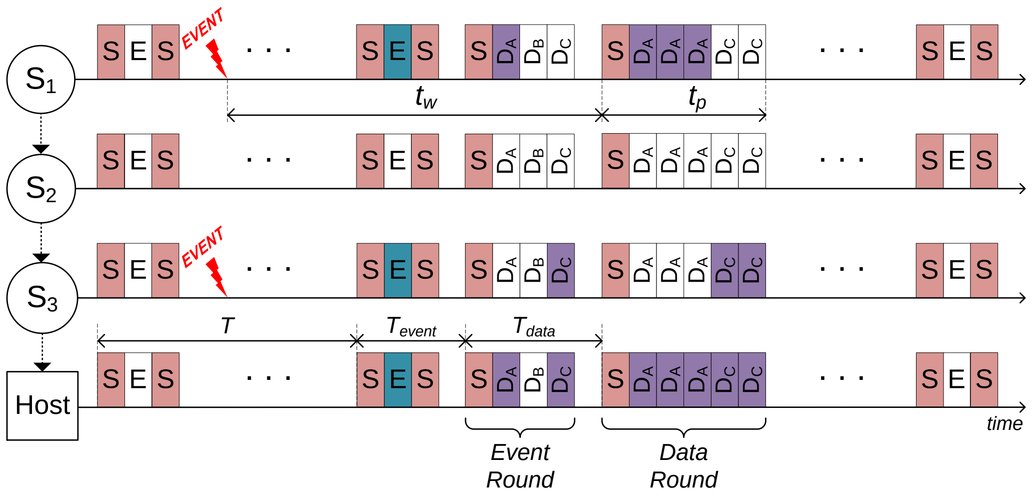


Figure 3.10: Example operation of the Event-based Low-power Wireless Bus (eLWB). Source nodes S_1 and S_3 detect an event within the same round period, and therefore both initiate a flood in the event contention slot. The host schedules an event round with contention-free slots for disseminating the event and the bandwidth requirements of the associated data. The host then schedules a data round with contention-free slots for the dissemination of the data.

and request bandwidth for their data stream. The schedule slot associated with the event round defines the time until the start of the next round, T_{data}, which informs all nodes of the beginning of the *data round*.

Once the host collects all the data stream requests, it partitions the available network bandwidth accordingly. In the example depicted in Figure 3.10, the host allocates data slots to nodes S_1 and S_3 according to their demands. The schedule slot of the data round defines the allocation of data slots to the respective source nodes, and specifies the new round period $T - T_{data} - T_{event}$. In order to facilitate the rapid dissemination of event and data streams, the time offsets of the event round T_{event}, and data round T_{data}, are chosen to be significantly less than the round period T.

While we have only considered until now the dissemination of events from source to host, it is important to highlight that the eLWB supports bi-directional dissemination of events and periodic data streams. A source node may utilize the event round to indicate a periodic stream, e.g., for node health information, and the host will schedule the corresponding data slot during each round. Additionally, the host may allocate data slots to facilitate unicast or broadcast communication between host and source nodes.

Protocol Limitations. Since the event round provides each source node with a dedicated data slot, the duration of the event round increases linearly with the number of nodes in the network. However, as each source node is only allocated one byte in the event round, the overhead remains realistic for typical deployments where the available bandwidth, $T - T_{data} - T_{event}$, constrains the number of simultaneous event disseminations to a small number of nodes, e.g., around 20 nodes. If larger networks are required, hierarchical structures may be employed, possibly in combination with non-overlapping communication channels.

As with all synchronous protocols, the eLWB exhibits a fundamental trade-off between end-to-end latency and energy efficiency. In order to achieve a lower event latency, the round period T must be reduced, resulting in an increase of energy consumption during periods of inactivity. However, as we experimentally evaluate in Section 3.4 using an indoor testbed, the proposed eLWB protocol achieves a best-case event latency of 113.2 ms, while dissipating on average as low as 49.7 μW.

3.3.4 Physical Platform Architecture

Using the system design methodology introduced in Section 3.2, we designed and implemented each event-triggered logical component such that the design constraints of responsiveness, energy efficiency and adaptability are adhered to. We now propose to integrate these logical components onto a physical platform architecture while preserving these same properties. We argue that in order to achieve this, the physical platform architecture must support the following:

- ***Limited Resource Interference.*** Components that are active concurrently may exhibit resource interference with respect to processor clock cycles, memory or peripherals. Such resource interference may adversely impact the performance of components, for example, where a component is prevented from entering a low-power sleep mode due to the concurrent processing of another component. In order to preserve the responsiveness and energy-efficiency of components, we must limit resource interference wherever possible.

- ***Composable Construction.*** In order to retain the properties of each event-triggered logical component, the platform architecture must support composability [Jan06]. This well-established system design principle makes it possible to interconnect components together without changing the properties, i.e., responsiveness and energy efficiency, of the integrated parts. This powerful property is facilitated by the interconnection of components using interfaces with formally defined semantics.

Proposed Platform Architecture. We achieve these two requirements by *(i)* mapping components that encounter resource interference onto

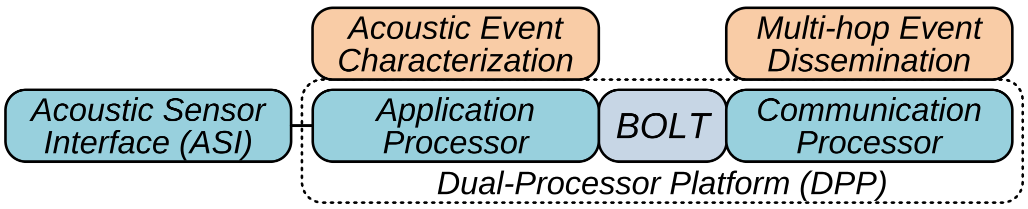
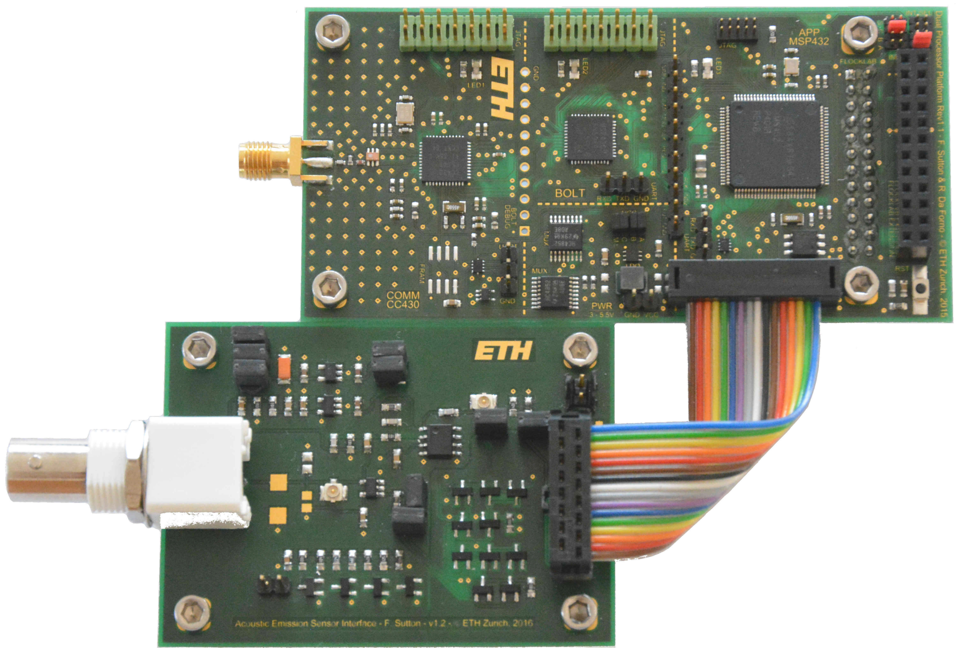

Acoustic Event Characterization
Multi-hop Event Dissemination
Acoustic Sensor Interface (ASI)
Application Processor
BOLT
Communication Processor
Dual-Processor Platform (DPP)

Figure 3.11: Physical platform architecture of the prototype wireless acoustic emission sensing system using the Bolt ultra-low power processor interconnect.

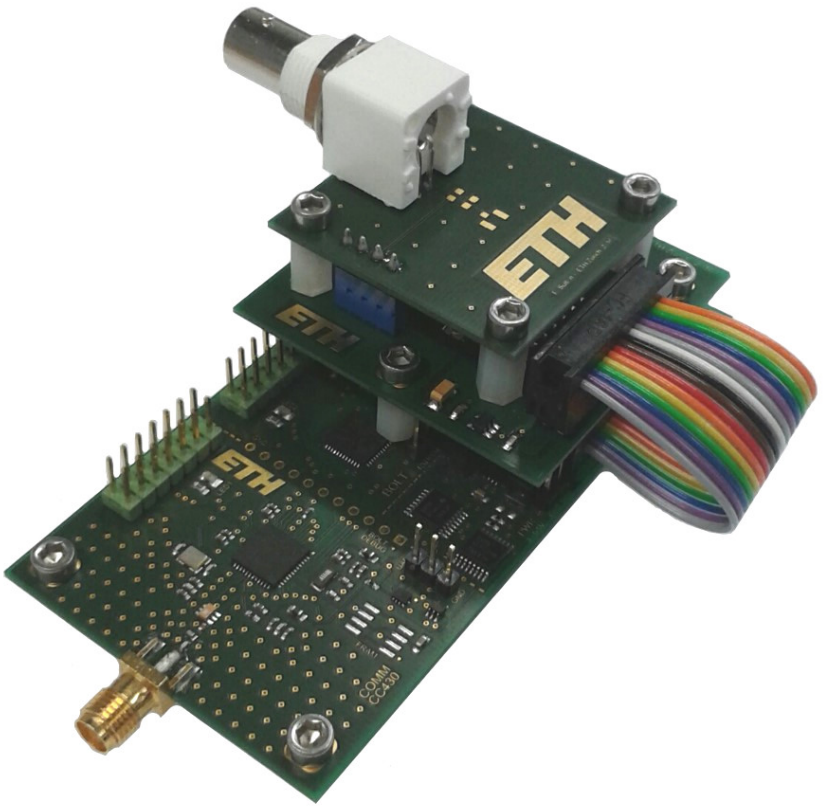
dedicated processors, and *(ii)* interconnect the processors using an well-defined interface with predictable run-time behavior. We propose to map the acoustic event characterization component to a dedicated application processor, and the multi-hop event dissemination component to a dedicated communication processor, as depicted in Figure 3.11. Since the acoustic sensor interface component consists of analog and digital components, it is interfaced directly to the application processor using GPIO lines. The application and communication processors are interconnected using Bolt, which as presented in Chapter 2, is an ultra-low power processor interconnect supporting bi-directional asynchronous message passing with predictable run-time behavior. Bolt decouples the two processors with respect to time, power and clock domains, thus enabling the processors to execute concurrently without risk of resource interference, while also facilitating composable integration of event-triggered components.

A prototype of the proposed platform architecture, termed the *Dual-Processor Platform* (DPP) together with the acoustic sensor interface are depicted in Figure 3.12(a). The wireless embedded platform consists of a 32-bit MSP432P401R ARM Cortex-M4 application processor running at 48 MHz, which is interconnected by Bolt to a 16-bit CC430F5147 communication processor running at 13 MHz. A fully-integrated prototype of the wireless acoustic emission sensing system is illustrated in Figure 3.12(b).

Summary. We began this section by introducing a logical event-triggered component model that captures responsiveness, energy efficiency and adaptability design constraints. We then constructed an wireless acoustic emission sensing system from a pipeline of logical components, presented a concrete realization of each component, and demonstrated their integration onto a novel physical platform architecture. We next experimentally evaluate the developed prototype in the context of a real-world wireless acoustic emission sensing scenario.


APP
MSP432
BOLT
COMM
CC430
ETH
Acoustic Emission Sensor Interface - F. Sutton - v1.2 - © ETH Zurich, 2016

(a)



(b)

Figure 3.12: (a) Prototype implementation of the Dual-Processor Platform (top) and the acoustic sensor interface (bottom), and (b) a fully-integrated prototype of the wireless acoustic emission sensing system.

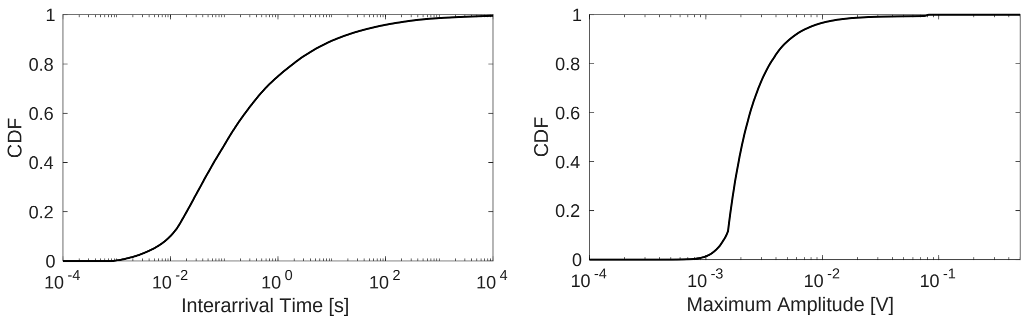


Figure 3.13: Cumulative distribution function of interarrival time and maximum amplitude of acoustic events extracted from a field deployment.

3.4 Case Study: Codetection of Acoustic Events

In this section, we consider a specific real-world application, the monitoring of acoustic emissions in steep fractured rock walls [GBG+12]. The goal is to identify rock damage and fracture propagation by detecting and characterizing acoustic emissions caused by cryogenic processes, e.g., volumetric expansion of freezing water within a rock wall. The deployment of a wireless acoustic emission sensing system makes it possible to capture these acoustic events with unprecedented spatial coverage and temporal resolution, which may in the future be used to develop early warning systems.

A field experiment spanning several months using a piezoelectric transducer installed 10 cm below the surface of a rock wall at 3500 m a.s.l [GBG+12] has identified these acoustic events to be sporadic and often occur in bursts. As depicted in Figure 3.13, the observed acoustic events have interarrival times between a few milliseconds and several hours, and exhibit a maximum amplitude from a few millivolts up to one hundred millivolts.

Given the severe implications of rockfall and the harsh deployment conditions, a wireless sensor deployment must *(i)* rapidly detect, characterize and communicate acoustic events for analysis, and *(ii)* maximize operational lifetime. It follows that the system design and implementation of the wireless acoustic emission sensing system presented in Section 3.3 not only satisfies these requirements by design, but also supports the application-specific requirements.

In order to demonstrate the suitability of the developed prototype to this real-world application, we experimentally evaluate the prototype in a *codetection* use case. This is a particularly challenging scenario where several wireless acoustic sensors detect and characterize the same acoustic event, before each node simultaneously disseminates the event through the network as rapidly and energy efficiently as possible. To this

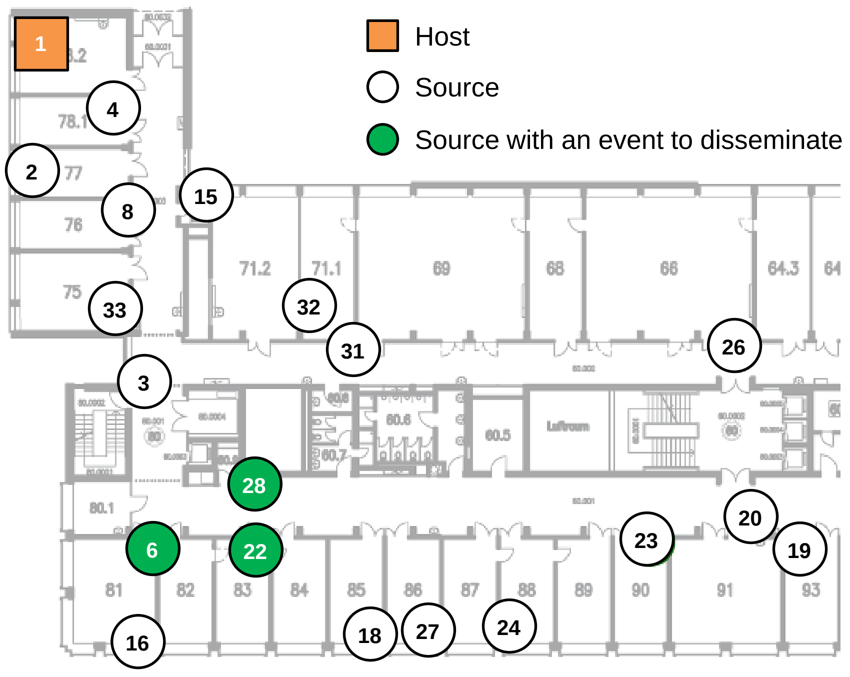


Figure 3.14: Map of the FlockLab indoor testbed deployment.

end, we first assess the responsiveness of the eLWB protocol using an indoor testbed, and then evaluate the power dissipation of the developed prototype while being triggered by a real-world acoustic signal.

3.4.1 Responsiveness of Multi-hop Event Dissemination

Setup. We emulate the codetection of acoustic events at the network-level by utilizing the FlockLab [LFZ+13b] indoor testbed configured with fine-grained tracing capabilities [LMD+15]. We deployed 20 Olimex MSP430-CCRF nodes programmed with the eLWB according to the map illustrated in Figure 3.14. We configured FlockLab to periodically trigger source nodes 6, 22 and 28 in order to emulate the simultaneous detection of an acoustic event during each eLWB round, therefore forcing the three source nodes to initiate a flood in each eLWB event contention slot.

We evaluate the performance of the eLWB using three metrics: *(i) event detection* is the number of event contention slot transmissions that are successfully received by the host, *(ii) event and data dissemination* is the number of event and data streams transmitted by a source that are received at the host without error, and *(iii) event and data latency* is the time between a source transmitting in the event contention slot and the successful reception of the data stream at the host. We compute all metrics based on 100 eLWB rounds using a static protocol configuration, as listed in Table 3.3.

Table 3.3: eLWB parameters used in the FlockLab indoor testbed experiments.

Parameter	Description
$T \in [5, 10, 15]$ s	Round period of the eLWB
$T_{event} = 40$ ms	Time offset until the beginning of the event round
$T_{data} = 60$ ms	Time offset until the beginning of the data round
$Q = 16$ bytes	Number of bytes per data slot
$N_S = 3$	Maximum number of transmissions for schedule slots
$N_E = N_D = 2$	Maximum number of transmissions for all other slots

Results. The success rate of event detection for all three contending source nodes is 100%, as illustrated in Figure 3.15 (top). This means that despite the simultaneous event contention slot transmissions, the host successfully identifies that at least one event must be disseminated each round. In order to evaluate a lower-bound performance, the host only provides three data slots in the event round, during which each node requests two data slots. The experimental results show that all three source nodes disseminate their respective event and data streams with a success rate above 98%.

The event and data latency represents the best-case delay between an event detection and the successful dissemination of both event and data streams. We evaluate this metric for both eLWB and LWB protocols for each round, with the average presented in Figure 3.15 (bottom). Firstly, we highlight that the latency per source node using the eLWB is approximately constant for all three round periods. This is the expected and desired behavior, since the operation of the event and data rounds are independent on the round period. Secondly, the latency of each source node, i.e., 113.2 ms for node 6, 142.8 ms for node 22, and 169.8 ms for node 28, differs only by the duration of approximately two data slots. This is to be expected in our implementation, as the host schedules two data slots within the data round and assigns them in the order the of source node identities, i.e., node 6 is assigned the first two, while node 28 is assigned the last two data slots.

In order to highlight the superior responsiveness of the eLWB protocol under contention, we compare against the original LWB. We adjust all LWB configuration parameters in order to improve its performance with respect to latency. Specifically, we set the LWB maximum and minimum round periods to $T_{max} = T$, and $T_{min} = 1$ s, respectively, and request a stream with a negative start time and an inter-packet interval of T_{min} a total of 40 times. Furthermore, and most importantly, we analyze the LWB event dissemination for only one source, i.e., node 6. This scenario represents the absolute best-case scenario for the LWB, as simultaneous

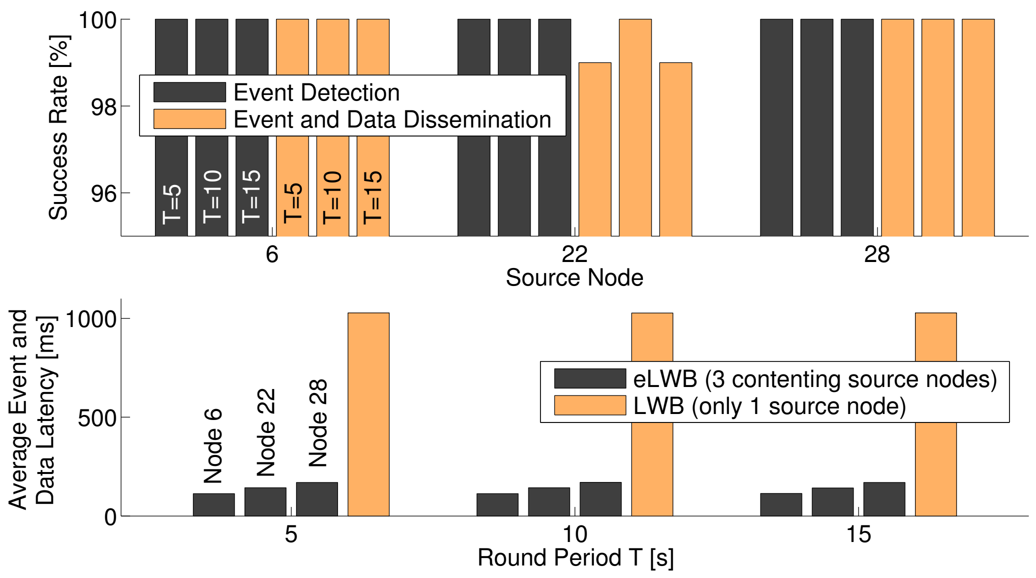


Figure 3.15: (top) Event contention and dissemination metrics of the eLWB, and (bottom) the average event and data latency of the eLWB compared to the LWB.

stream request transmissions would undoubtedly invoke random back-off, thereby delaying event dissemination by multiples of T_{min}. In summary, the experimental results show that the mean event latency of the eLWB is, at the very least, five times better than the LWB, whilst providing reliable dissemination for the codetection of acoustic events.

3.4.2 Power Dissipation of Developed Prototype

We first experimentally evaluate the power dissipation of the developed prototype under a static configuration, before using the event-triggered component model presented in Section 3.2 to estimate the average power dissipation of the system with alternative run-time configurations.

3.4.2.1 Measurement of Static Configuration

Setup. We measure the power dissipation of the developed prototype arranged in a single-hop wireless network using an Agilent N6705A DC power analyzer at a supply voltage of 2.5 V. The source node is allowed to reach a steady operational state by synchronizing itself to the periodic eLWB rounds initiated by the host. We emulate a real-world acoustic signal by connecting an Agilent 33600A arbitrary waveform generator to the input of the acoustic sensor interface and playback an acoustic signal extracted from the field.

Results. Figure 3.16 illustrates the power dissipation of the prototype source node during periods of inactivity and during the detection,

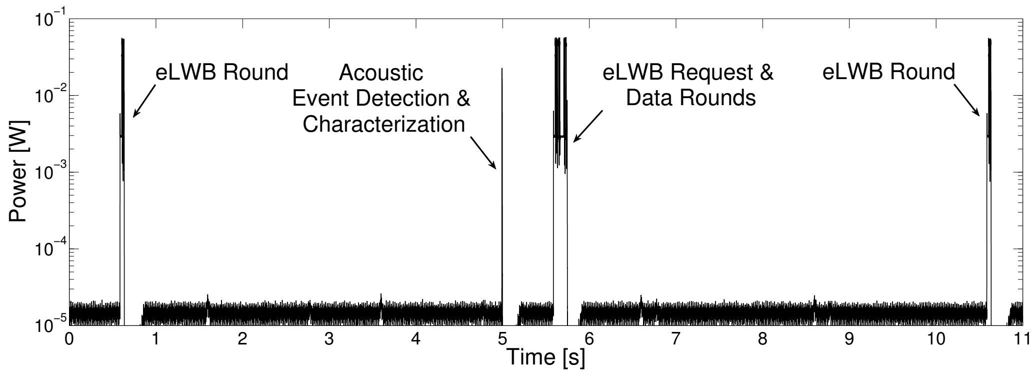


Figure 3.16: Power dissipation of the developed prototype detecting and characterizing an acoustic event extracted from a real-world application scenario, before dissemination using the eLWB protocol configured with $T = 5\,\mathrm{s}$.

Table 3.4: Power dissipation of each component during sleep state. The developed prototype exhibits a total power dissipation of 59.7 μW during periods of inactivity.

Component	$\mathrm{P_s}$
Acoustic Sensor Interface (v1.2)	6.2 μW
Acoustic Event Characterization	2.5 μW
Bolt Processor Interconnect	1.3 μW
Multi-hop Event Dissemination ($T = 15\,\mathrm{s}$)	49.7 μW

characterization and dissemination of an acoustic event. Approximately half a second into the experiment, the multi-hop dissemination component awakes from sleep mode and checks if there are any pending messages in Bolt. As there are no messages, the component participates in the eLWB round without initiating a transmission in the event contention slot, and returns back to sleep. At approximately five seconds into the trace, a real-world acoustic signal is injected into the acoustic sensor interface, awaking it from sleep, and subsequently triggering the acoustic event characterization component. Once all features have been extracted from the digitized acoustic signal, a message is written into Bolt containing the event feature set. The next eLWB round begins approximately half a second later, and the pending message is read out from Bolt, thus triggering multi-hop event dissemination. The source node indicates an event by initiating a transmission during the event contention slot and proceeds to disseminate the event using the event and data rounds according to the eLWB protocol detailed in Section 3.3.3.2, before returning to sleep. As there are no further acoustic events emulated, the source node awakes five seconds later for the next eLWB round.

The power dissipation of each component of the developed prototype

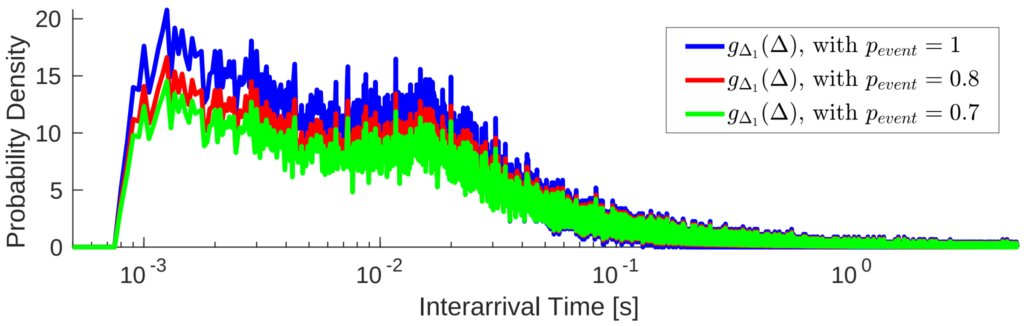


Figure 3.17: Probability density of input acoustic events for alternative run-time configurations of the acoustic sensor interface.

Table 3.5: Average power dissipation of system components with alternative run-time configurations of the acoustic sensor interface.

Component	**Average Power Dissipation P_{avg}**		
	$p_{event} = 1$	$p_{event} = 0.8$	$p_{event} = 0.7$
Acoustic Sensor Interface (v1.2)	38.5 μW	31.9 μW	28.7 μW
Acoustic Event Characterization	150.8 μW	125.1 μW	112.4 μW
Multi-hop Event Dissemination	3.6 mW	3.0 mW	2.7 mW

during sleep state is listed in Table 3.4. The developed prototype dissipates a total of 59.7 μW during sleep state, thus making it possible to support the codetection of acoustic events for more than one year using low-capacity coin cell batteries.

3.4.2.2 Estimation of Adaptive Configuration

In practice, a system designer is not only interested in the power dissipation of the system during periods of inactivity, but also during realistic operating conditions with alternative run-time configurations. In order to evaluate the impact of component-level adaptivity, we utilize the logical event-triggered component model presented in Section 3.2 to estimate the average power dissipation of the developed prototype.

Simulation Setup. We consider a scenario where the detection threshold for acoustic emissions is adjusted at run-time, and is done so without imposing any energy overhead, i.e., $P_f = 0$. We randomly filter an acoustic event stream extracted from the field to produce the probability density $g_{\Delta_1}(\Delta)$ for a set of p_{event} values, as illustrated in Figure 3.17. The choice of p_{event} increases the average event interarrival time, thereby decreasing the event arrival rate β, which is synonymous to increasing the detection threshold. We then estimate the average power dissipation P_{avg} of each component according to the analytical framework presented in Section 3.2.

Results. Table 3.5 lists the average power dissipation P_{avg} of each component for three run-time configurations. As expected, the power dissipation of each component decreases as the p_{event} decreases since the random filtering of events reduces the average rate at which acoustic events are detected by the system. It is evident from the analysis that the difference in power dissipation of the multi-hop event dissemination component between run-time configurations is up to two orders of magnitude larger than the power dissipation of the other two components.

3.5 Related Work

The PinPtr [SML+04] sniper detection system, the volcanic monitoring system presented in [WAJR+05] and the environmental monitoring system in [GBG+12] all perform continuous sampling of an acoustic sensor, which leads to limited operational lifetime or necessitates appropriately dimensioned energy harvesting capabilities. We present a solution based on sensor-initiated wake-up, ensuring event characterization is performed only when an event actually occurs, therefore reducing energy consumption during periods of inactivity.

CargoNet [MMF+07] and the structural monitoring system presented in [LHV+09] are the closest to our work with respect to sensor-based wake-up and multi-hop data dissemination. The node used in CargoNet also employs an ultra-low power comparator to detect events but relies on an RFID transponder to initiate the communication of stored measurements. The sensing platform presented in [LHV+09] continually samples a strain gauge at 100 Hz in order to start the acquisition and signal processing of its attached acoustic sensors, before the data is transmitted using Low-Power Listening [PHC04]. Our solution not only integrates an ultra-low power acoustic interface, but also incorporates a responsive and energy efficient wireless protocol that supports dissemination when an event is simultaneously detected by multiple nodes.

Zahedi et al. [ZYH15] present a passive wireless sensing system where an acoustic emission signal is directly modulated onto a high-frequency carrier. A high-powered reader generates the high-frequency carrier, and demodulates the received signal in order to recover the acoustic emission signal. While this solution is a very elegant single-hop solution between sensor and reader, it does not easily scale to a multi-hop network scenario due to limitations in communication range and challenges associated with concurrent transmissions.

3.6 Summary

In this chapter, we detail the challenges associated with the design and implementation of efficient event-triggered wireless sensing systems, and propose a novel design methodology to guide the system designer through the component-level trade-offs that lead to an efficient realization. We model the end-to-end sensing system as a pipeline of event-triggered logical components, where each logical component encapsulates responsiveness, energy efficiency and adaptability design constraints. Each component is concretized in isolation using well-known design techniques before all components are integrated onto a physical platform architecture that preserves the underlying design constraints. We exemplify our approach through the prototype development of a wireless acoustic emission sensing system, which is experimentally evaluated in terms of responsiveness and energy efficiency. The developed prototype is capable of detecting an acoustic event with a delay of 16 μs, starting to characterize an acoustic event with a delay of 29 μs, and disseminating an event and its associated data over a multi-hop network under contention with an average latency as low as 113.2 ms, while only dissipating a total of 59.7 μW during periods of inactivity. The presented prototype is therefore able to capture the dominant spectral characteristics of acoustic emissions relevant to a range of application domains, while supporting multi-year operation using low-capacity batteries.

4 On-demand Network Flooding

The dissemination of periodic events through a multi-hop network is an application scenario that has been well-studied in the wireless sensor network community, and as a result, many energy-efficient protocols have been proposed and experimentally evaluated. These solutions are typically based on radio duty-cycling, where the achievable end-to-end latency of the multi-hop dissemination is proportional to the periodicity of the radio communication. However, there are many application scenarios where events do not occur periodically, but only on rare occasions. Therefore, in order to disseminate a rare event with low end-to-end latency using a radio duty-cycled protocol, nodes must consume significant energy on superfluous communications.

In this chapter we present Zippy, an asynchronous protocol that supports low-latency multi-hop dissemination of rare events while dissipating several orders of magnitude less power compared to state-of-the-art duty-cycled protocols. Zippy takes advantage of low-complexity radio receivers coupled with network flooding techniques to facilitate on-demand asynchronous network wake-up, fine-grained per-hop synchronization, and efficient multi-hop dissemination of an event packet to a remote host. We present the design and analysis of Zippy before detailing a prototype implementation using a custom-built wireless embedded platform constructed from commercially available components. We extensively evaluate Zippy's performance in a laboratory setting and in an indoor testbed. Zippy achieves on-demand flooding of rare events with an end-to-end latency on the order of tens of milliseconds, a per-hop synchronization on the order of tens of microseconds, and a power dissipation of 9.6 μW during periods of inactivity.

4.1 Introduction

Motivation. The past 15 years of wireless sensor network research has produced an abundance of energy-efficient protocols for disseminating *periodic* events through a multi-hop network of resource-constrained wireless sensor nodes, as surveyed in [HXS+13]. However, there are many real-world applications, such as industrial automation [GRK10], structural monitoring [KPC+07], medical alert systems [PJ09], surveillance systems [ZFB10], and environmental monitoring [GBG+12], where events are not periodic, but occur only on *rare* occasions. Under this premise, state-of-the-art duty-cycled protocols are faced with an undesirable design trade-off between energy efficiency and end-to-end latency.

In this chapter, we demonstrate a change in low-power protocol design, where the aforementioned design trade-off is circumvented. This enables low-latency dissemination of rare events through a multi-hop network to a remote host with unprecedented energy-efficiency.

Challenge. The key challenge is how to disseminate a rare event through a multi-hop network with low end-to-end latency, while doing so in an energy-efficient manner. In order to exemplify the design trade-offs encountered when applying state-of-the-art techniques, we consider a concrete application scenario, whereby resource-constrained wireless gas sensors must rapidly alert a remote host when the measured gas concentration exceeds a safe level [CEP16]. One approach is to apply a synchronous protocol, such as S-MAC [YHE02], T-MAC [VDL03], DW-MAC [SDGJ08] or Glossy [FZTS11], to disseminate the event through the network to the host. However, since the periodicity of the radio activity is directly linked to the end-to-end latency, all nodes in the multi-hop network must expend significant energy in order to achieve a low reporting latency of the event.

An alternative is to employ a pseudo-asynchronous protocol [LRW04], such as sender-initiated protocols B-MAC [PHC04] or X-MAC [BYAH06], receiver-initiated protocols Koala [MELT08] or A-MAC [DDHC+12], hybrid protocols TRAMA [ROGLA03] or Z-MAC [RWA+08], or run-time adaptive protocols pTunes [ZFM+12] or Staffetta [CLZ+16]. However, as demonstrated in [DGA+05], radio duty-cycled protocols suffer from the same undesirable design trade-off. That is, the need for low end-to-end latency increases the radio activity at each node in the network, thereby reducing the operational lifetime of each node.

Yet another alternative is to employ backscatter technology, where a dedicated [ZHPG14] or an ambient [LPT+13] high-power infrastructure provides nodes with a medium for wireless communication. However, such techniques are limited in their support for energy-efficient multi-hop connectivity due to the energy costs of generating the high-power infrastructure signals. Furthermore, ambient infrastructure may not always be available or the achievable per-hop communication range is insufficient.

Approach. We present Zippy, a novel flooding technique that circumvents the aforementioned design trade-off by taking advantage of low-complexity transmitter and receiver hardware. We exploit the ultra-low power dissipation and unique timing properties of low-complexity radio hardware to perform on-demand flooding of rare events featuring end-to-end latencies on the order of milliseconds, while dissipating less than 10 μW during periods of inactivity. This level of energy efficiency makes it possible to support node operation for multiple years using low-capacity coin cell batteries.

Zippy combines four extensible components, which facilitate its adoption to a wide range of application domains. Firstly, robust *asynchronous network wake-up* ensures that all nodes in the multi-hop network are awoken from a low-power sleep state. Secondly, novel *neighborhood synchronization* achieves tight per-hop synchronization. Thirdly, efficient bit-level *event dissemination* supports the dissemination of event packets with near constant end-to-end latency. Finally, *carrier frequency randomization* ensures that the destructive interference associated with concurrent transmissions are mitigated during Zippy floods.

Contributions. This chapter makes the following contributions:

- We present an energy-efficient scheme for performing on-demand wake-up of a multi-hop network using low-complexity radio hardware.

- We introduce a novel technique for per-hop synchronization and flooding of an event packet through a multi-hop network.

- We introduce the design of an ultra-low power wireless embedded platform supporting Zippy, which dissipates only 9.6 μW during periods of inactivity.

- We detail a prototype implementation of Zippy, and extensively evaluate its performance both with cabled laboratory experiments and within a wireless indoor testbed.

4.2 Asynchronous Rendezvous

Zippy exploits an asynchronous rendezvous mechanism to facilitate on-demand network flooding. In this section, we study how a node can achieve on-demand rendezvous with its one-hop neighbors using low-complexity radio hardware.

4.2.1 Background

One of the fundamental challenges of low-power wireless communication is how to support energy-efficient packet exchange between single-hop neighbors, while minimizing the energy consumed due to idle listening. Given two arbitrary wireless nodes within range, packet exchange is only supported when both nodes have their radios turned on at the same time, which is termed a rendezvous [LRW04]. If a node does not know when its neighbor wants to send a packet, it must expend energy by powering its wireless receiver in anticipation of packet exchange. This behavior is termed idle listening [DEA06], and is a significant source of energy waste due to the high-power dissipation of commonly used receivers.

Wireless sensor network research has primarily addressed the problem of idle listening in the *time* domain [STGS02]. That is, by duty cycling the radio at the appropriate time, rendezvous between one-hop neighbors is supported, while also reducing the average energy consumption of all nodes in the network. However, under such duty-cycled schemes, there exists a fundamental trade-off between energy efficiency and how often the nodes rendezvous. Either the radio duty-cycle is increased to achieve a high rendezvous rate at the cost of higher energy consumption, or the radio duty-cycle is decreased to reduce energy consumption at the cost of a low rendezvous rate. In the case where on-demand rendezvous is desired, e.g., to rapidly exchange a rare event, a design decision must be made between the energy efficiency of the node and the latency of the event dissemination.

An alternative approach is to address the problem of idle listening in the *power* domain. Instead of duty cycling the radio hardware, the power dissipation of the radio is such that it becomes feasible to have it always turned on. There are two variants of this approach, based on the dependency on *infrastructure*, or not.

Backscatter technologies are an *infrastructure-based* approach, where a high-power signal source, i.e., either dedicated in the case of RFID tags [ZHPG14] or ambient in the case of ambient backscatter [LPT$^+$13], provides a medium in which passive RFID tags or backscatter-enabled devices can communicate by reflecting an incident signal. Despite significant advancements [PLGS15, KPG$^+$15, BJKK15, ZBJK16, VHPP$^+$17],

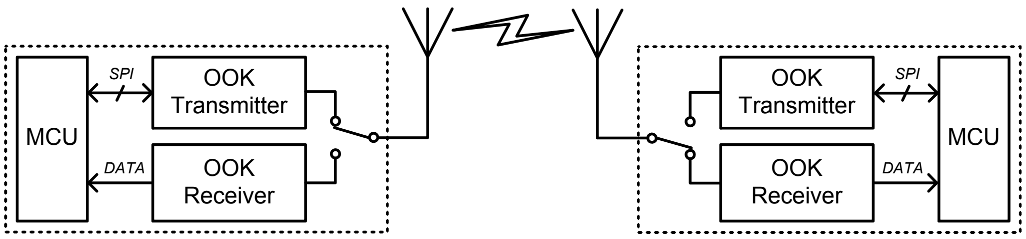


Figure 4.1: Proposed node architecture with a microcontroller (MCU), OOK transmitter, and always-on ultra-low power OOK receiver.

backscatter is limited in its support for energy-efficient multi-hop connectivity as either dedicated high-power carrier signals must be continuously generated, ambient infrastructure may not always be available, or the achievable per-hop communication range is insufficient.

We turn our attention to an *infrastructure-less* power domain solution to idle listening. Specifically, we use low-complexity receivers to decode transmissions from low-complexity transmitters. In contrast to backscatter technologies, the receiver is an always-on active circuit specifically designed for ultra-low power operation, while the transmitter is only used when there is an event to disseminate.

This particular approach has been proposed in the literature [GZR01] more than fifteen years ago, with the low-complexity receivers referred to as wake-up receivers or wake-up radios. However, recent advances in ultra-low power electronics and the commercial availability of components have made it feasible to achieve always-on idle listening while dissipating on the order of a few microwatts. This is in stark contrast to the order of tens of milliwatts dissipated by commodity radio hardware typically integrated into commercially available wireless embedded platforms.

We next present how such ultra-low power receiver structures are realized in practice.

4.2.2 Architecture for Asynchronous Rendezvous

The key to achieving ultra-low power on-demand rendezvous lies in the complexity of the radio hardware and the modulation scheme. Embracing low-complexity digital modulation schemes, such as On-Off Keying (OOK), drastically simplifies the receiver and transmitter circuitry, i.e., by reducing the number and specificity of components, which leads to a significant reduction in power dissipation. In particular, ultra-low power receiver designs, i.e., exhibiting sub-microwatt power dissipation, have been demonstrated in the literature for the acoustic [YKK13], radio frequency [PGR07], and optical [KLB+12] spectra.

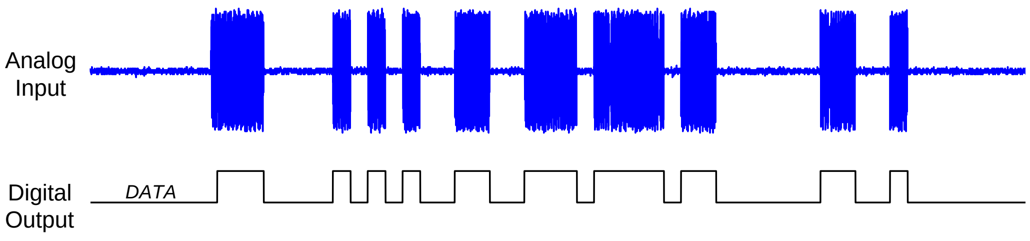


Figure 4.2: Example OOK signal at the input to the receiver and its corresponding output encoded on the *DATA* line.

We focus on low-complexity radio hardware for the ISM radio frequency band using the OOK digital modulation scheme. Figure 4.1 illustrates the node architecture assumed throughout this chapter, where each node consists of a microcontroller (MCU) provisioned with an OOK transmitter and an always-on OOK receiver. An RF switch controlled by the microcontroller selects the RF path from the antenna to either the input of the receiver, or the output of the transmitter. The OOK transmitter is designed such that its power dissipation is less than that of typical transceivers used for higher-complexity digital modulations, e.g., the MAX7044 features a current drain of 14 mA compared to the CC430 which drains 29 mA with identical output transmission power. The OOK receiver is designed such that its power dissipation is on the order of microwatts, i.e., several orders of magnitude below typical transceivers, thereby allowing always-on operation.

On-Off Keying is attributed as being the digital modulation scheme with the lowest complexity [Oet79], where a 1 bit is represented by the presence of a carrier frequency over a bit period T_b, while a 0 bit is represented by the absence of a carrier frequency over a bit period T_b. Figure 4.2 illustrates an example OOK-modulated bit sequence, both in its analog representation observed at the antenna of the OOK receiver, and its digital representation after decoding. The OOK receiver provides a single digital output, a *DATA* line, which is interfaced directly to the attached microcontroller for processing, which will be discussed next.

As illustrated in Figure 4.2, the OOK receiver manipulates the digital level of the *DATA* line according to the envelope of the received OOK signal. Since the node's microcontroller observes the state of the *DATA* line, it can perform two important functions, namely, *(i)* facilitate asynchronous rendezvous by the detection of the first rising edge of the *DATA* line, and *(ii)* decode an OOK modulated bit sequence by sampling the state of the *DATA* line within each bit period T_b. As discussed in Section 4.3, these two functions provided by the OOK receiver are fundamental to the operation of Zippy.

Despite the superior power-efficiency of low-complexity OOK

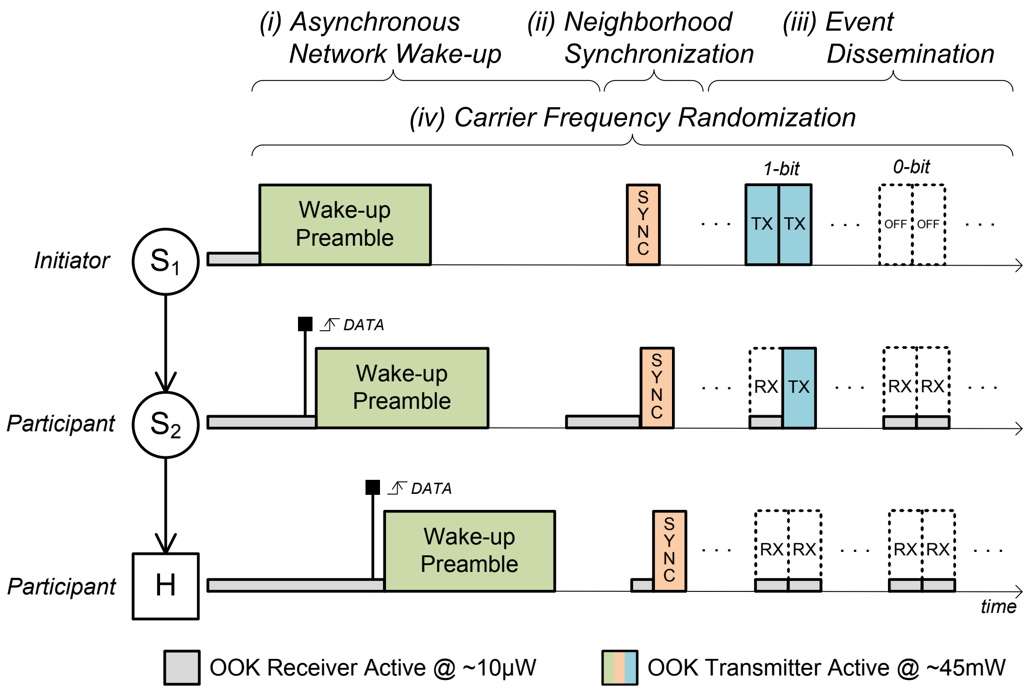

(i) Asynchronous Network Wake-up
(ii) Neighborhood Synchronization
(iii) Event Dissemination
(iv) Carrier Frequency Randomization
1-bit
0-bit
Initiator
S_1
Wake-up Preamble
SYNC
TX TX
OFF OFF
DATA
Participant
S_2
Wake-up Preamble
SYNC
RX TX
RX RX
DATA
Participant
H
Wake-up Preamble
SYNC
RX RX
RX RX
time
OOK Receiver Active @ ~10µW
OOK Transmitter Active @ ~45mW

Figure 4.3: Overview of Zippy in an example 2-hop network, where the initiator node has a rare event to disseminate through the multi-hop network.

receivers, they exhibit well-known practical limitations, as surveyed in [DEO09]. We systematically evaluate the two most important limitations in Section 4.5.

We next describe how asynchronous rendezvous using low-complexity radio hardware is used in a multi-hop network to support tight per-hop time synchronization and energy-efficient bit-level data dissemination.

4.3 Overview of Zippy

In this section, we present an overview of Zippy, an asynchronous protocol that demonstrates the feasibility of on-demand multi-hop flooding using low-complexity radio hardware. Figure 4.3 exemplifies the operation of Zippy, where an *initiator*, node S_1, disseminates a rare event to the remote host H through a 2-hop network consisting of *participant* nodes S_2 and H. We next briefly describe the function of Zippy's four extensible components.

Asynchronous Network Wake-up. The initiator commences a Zippy flood by transmitting a wake-up preamble, i.e., a sequence of 1 bits, using its low-complexity OOK transmitter. Since the OOK receiver at node S_2 is always on, the reception of the wake-up preamble will assert the receiver's *DATA* line, awaking its microcontroller from a low-power sleep mode. Node S_2 will then proceed to transmit a wake-up preamble so to wake-up its neighbor, node H. The relaying of the wake-up preamble continues until all nodes in the network are awake, and as a consequence, all nodes achieve coarse-grained time synchronization, i.e., on the order of milliseconds, to their nearest one-hop neighbors.

Neighborhood Synchronization. Shortly after the completion of the wake-up preamble transmission, the initiator transmits a single OOK 1 bit, termed the synchronization bit. Once the receiver of node S_2 detects the beginning of this bit, i.e., by the rising edge of the receiver's *DATA* line, the node immediately transmits a synchronization bit. The same process occurs at node H. The time delay between the beginning of the synchronization bit transmission, and the instant it is detected at the microcontroller provides fine-grained per-hop time synchronization, i.e., on the order of tens of microseconds.

Event Dissemination. All nodes now take advantage of the per-hop time synchronization to propagate a packet through the network with bit-level granularity. The packet is encoded using a repetition code, with the length of the code equal to the maximum hop count of the network. In the considered example, each bit of the packet is represented by two sub-bits, i.e., a 1 is transmitted as 11 and a 0 is transmitted as 00. The initiator transmits all packet sub-bits, while each participant uses its OOK receiver to decode each sub-bit in sequence. If a sub-bit is decoded as a 1, the participant switches to its OOK transmitter and transmits any remaining 1 sub-bits, thereby propagating each bit to its neighbors.

Carrier Frequency Randomization. It is evident that transmissions during a Zippy flood will overlap. As shown in [WLS14], the packet reception rate reduces as the number of concurrent transmissions increases. Furthermore, since we use low-complexity OOK receivers, we are not able to take advantage of the capture effect [LF76], which is associated with high-complexity receivers. Although it is shown in [Ash92] that OOK receivers may be designed to exhibit the capture effect, the additional hardware increases the total power dissipation of the receiver, potentially negating the benefits of using it for asynchronous rendezvous. Therefore, during the execution of all aforementioned components of Zippy, we mitigate the impact of overlapping OOK transmissions by randomizing the carrier frequency used to represent OOK 1 bits.

While the node architecture assumed throughout this chapter targets the ultra-low power and low-complexity boundaries of the radio sub-system design space, we do not restrict the adoption of Zippy's four

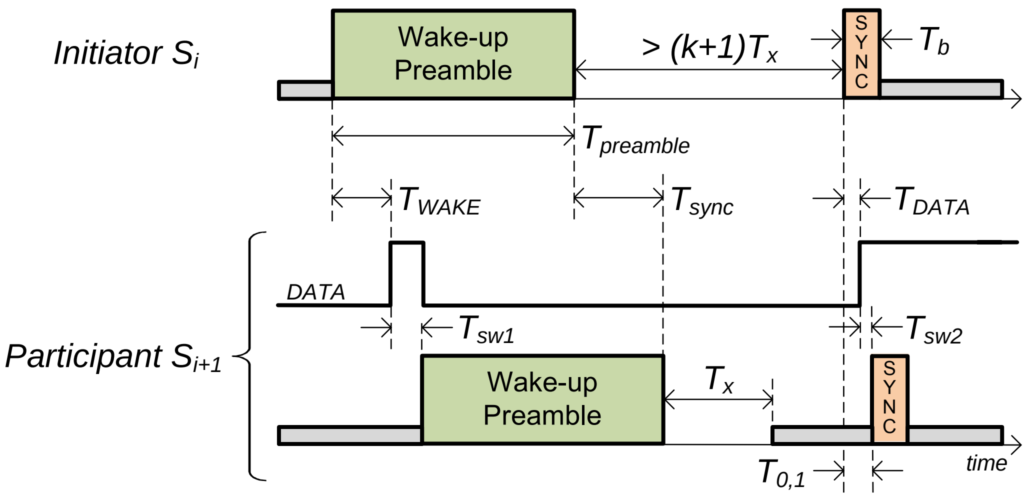


Figure 4.4: Timing diagram for Zippy's asynchronous network wake-up and neighborhood synchronization.

extensible components into existing wireless embedded platforms and/or protocols. On the contrary, the fundamental components of Zippy may be combined to create innovative platform architectures and/or energy-efficient extensions to established protocols. For example, asynchronous network wake-up and carrier frequency randomization provide a widely applicable and robust multi-hop wake-up service, while a state-of-the-art flooding protocol may benefit from the integration of asynchronous network wake-up, neighborhood synchronization and carrier frequency randomization components in order to reduce the energy footprint of disseminating events and their associated data. A concrete example of such communication architecture is presented in Chapter 5.

4.4 Design and Analysis of Zippy

We next present the design and analysis of Zippy's four extensible components. Where applicable, we characterize certain properties of the OOK receiver so to reinforce the design and analysis of each component.

4.4.1 Asynchronous Network Wake-up

A precondition for communicating with all nodes in a network is to ensure they are all turned on and listening, i.e., the node's microcontroller is active with a radio interface configured in receive mode. With the assistance of an always-on low-complexity OOK receiver integrated in each node, the action of remotely waking-up a node is achieved by the transmission of a wake-up preamble, i.e., a fixed-length sequence of 1 bits.

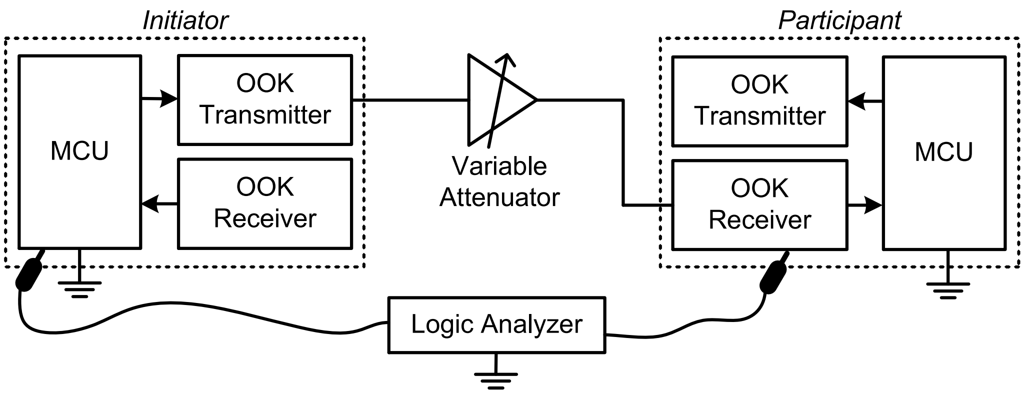


Figure 4.5: Experimental setup for the timing analysis of the OOK receiver's *DATA* line using a logic analyzer.

As introduced in Section 4.2, the always-on ultra-low power OOK receiver will detect the beginning of the wake-up preamble, activate its *DATA* line, and bring the attached microcontroller out of a low-power sleep mode using appropriate interrupt processing.

It follows that the fastest way to facilitate node wake-up throughout a multi-hop network is to relay the wake-up preamble along each hop. That is, as soon as the *DATA* line is asserted by the OOK receiver, the attached microcontroller awakes from low-power sleep mode, and proceeds to transmit a wake-up preamble. The duration of the wake-up preamble $T_{preamble}$ must therefore be longer than the time taken for the OOK receiver to detect the wake-up preamble and assert its *DATA* line. Figure 4.4 illustrates the detailed timing relations between initiator and participant nodes. In order to determine a suitable duration of the wake-up preamble, we must measure the delay T_{WAKE} from the beginning of the initiator's wake-up preamble transmission until the rising edge of the participant's *DATA* line. This timing characteristic is dependent on two factors, namely, *(i)* the OOK receiver hardware, and *(ii)* the RF propagation effects. We next analyze the timing properties of the OOK receiver used throughout this chapter under ideal RF conditions in order to give a lower bound on the duration of the wake-up preamble.

Using the experimental setup illustrated in Figure 4.5, we measure the time taken for the receiver to assert its *DATA* line, T_{WAKE}, across the operating range of the receiver as controlled by a variable attenuator. A logic analyzer is used to measure the time between the start of the wake-up preamble transmission at the initiator's microcontroller, and the rising edge of the *DATA* line at the participant's OOK receiver. Figure 4.6 plots the distribution of T_{WAKE} as the channel attenuation is varied such that the input signal power to the OOK receiver is between −31 dBm and −51 dBm.

The results indicate a very stable time distribution for T_{WAKE} across the

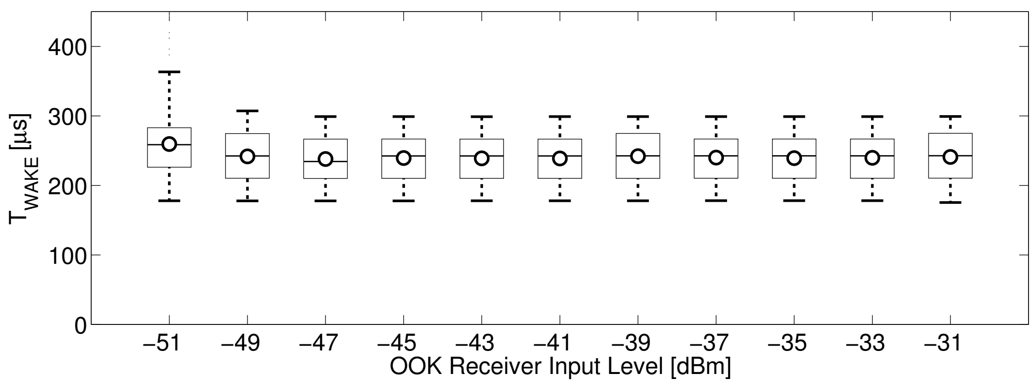


Figure 4.6: Delay to trigger the rising edge of the *DATA* line with respect to the receiver's input signal level.

operational range of the receiver. This is primarily due to the automatic gain control (AGC) stage integrated in the OOK receiver, as detailed in Section 4.5.1. The AGC continuously adjusts a variable-gain amplifier to ensure that low input signals are able to be successfully decoded. Based on these experimental results, the wake-up preamble must be at least as long as the maximum observed T_{WAKE}, i.e., $T_{preamble} > 370\,\mu$s.

Once the *DATA* line is asserted at a participant's microcontroller, the transmission of a wake-up preamble commences. The time taken to start the data radio transmission, T_{sw1}, is determined by the start-up time of the transmitter, and the software execution time needed to configure the hardware and fill the transmit buffer. This software delay can be made approximately constant, e.g., the implementation detailed in Section 4.5.1 exhibits a T_{sw1} of approximately 350 μs.

At the end of the asynchronous network wake-up, the microcontroller of each wireless node is awake, and has coarse-grained time synchronization to their nearest neighbor. Specifically, the per-hop synchronization, T_{sync}, is given by (4.1), where T_{prop} is the propagation delay of the carrier signal.

$$T_{sync} \geq T_{WAKE} + T_{sw1} + T_{prop} \tag{4.1}$$

Based on the aforementioned characterization of the OOK receiver, the per-hop synchronization will be within one millisecond. We next investigate how the per-hop time synchronization can be improved by further taking advantage of the low-complexity OOK receiver's timing properties.

4.4.2 Neighborhood Synchronization

Since per-hop time synchronization is a prerequisite for low-latency data dissemination through a multi-hop network, we aim to improve the per-hop time synchronization achieved by the asynchronous network

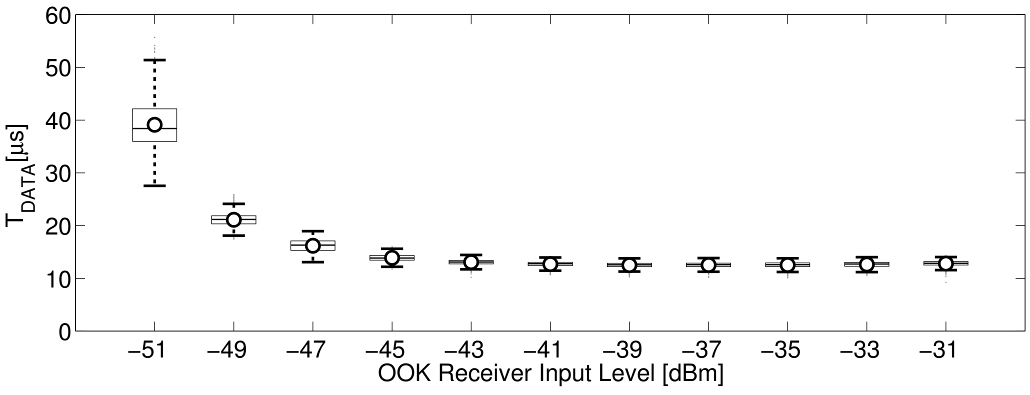


Figure 4.7: Delay to assert the *DATA* line in response to a neighbor's transmission of a 1 bit.

wake-up. We take advantage of the fact that low-complexity OOK receivers decode on a bit-level, as opposed to a byte-wise or packet-wise reception offered by high-complexity receivers, e.g., IEEE 802.15.4 compatible receivers. As introduced in Section 4.2, the OOK receiver provides a digital *DATA* line that represents the envelope of a received OOK signal. We now utilize the time it takes for the *DATA* line to assert, T_{DATA}, given that a neighboring node has transmitted a 1 bit, in order to tightly synchronize nodes to their neighbors. As indicated in Figure 4.4, the delay T_{DATA} is expected to be smaller than T_{WAKE} since the receiver's AGC has settled to a steady state after the reception of the wake-up preamble.

Using the experimental setup illustrated in Figure 4.5, we measure the time between the beginning of a 1 bit transmission from the initiator, until the rising edge of the participants *DATA* line using a logic analyzer. The time separating the wake-up preamble and the 1 bit transmissions is chosen to be longer than the receiver's AGC settling time. Figure 4.7 illustrates the box plot of the delay T_{DATA}, as the OOK receiver's input signal level varies between −31 dBm and −51 dBm. We observe a statistically stable delay with an average of approximately 13μs for low and medium signal strengths. The average delay increases by a factor of three as the input signal power approaches the receiver's sensitivity level of −52 dBm, as evaluated in Section 4.5.1.1.

We use this advantageous timing property of the OOK receiver to achieve per-hop synchronization on the order of tens of microseconds. Once the asynchronous network wake-up is complete, as depicted in Figure 4.4, the initiator transmits a synchronization bit, i.e., a single 1 bit of duration T_b, using its OOK transmitter. The participant's OOK receiver will detect this bit, and assert its *DATA* line after an elapsed time of T_{DATA}. The participant's microcontroller then begins to transmit a synchronization bit using its OOK transmitter. The time to switch on the OOK transmitter, T_{sw2}, is a combination of hardware and software delays which are considered constant.

The relaying of the synchronization bit continues throughout the multi-hop network. In an ideal RF propagation environment, the time between the start of the synchronization bit transmission at hop i, and the start of the synchronization bit transmission at hop $i+1$ is at least $T_{i,i+1}$ seconds, as given by (4.2). The constant T_{prop} represents the carrier signal propagation time, which is considered negligible.

$$T_{i,i+1} \geq T_{data} + T_{sw2} + T_{prop} \tag{4.2}$$

The aforementioned neighborhood synchronization scheme has two caveats that necessitate further discussion. Firstly, once a participant has completed its carrier burst transmission, it must wait a minimum of T_x seconds before turning on its receiver in preparation for the relaying of the synchronization bit. If the receiver is turned on too early, the *DATA* line will activate in response to a neighbors wake-up preamble transmission, causing erroneous synchronization. Therefore, each participant must wait at least $T_x = \max(T_{WAKE}) + T_{sw1}$ seconds before reacting to the rising edge of its receiver's *DATA* line.

Secondly, the initiator must ensure that the synchronization bit is not transmitted before all nodes within the multi-hop network have completed their wake-up preamble transmissions. Therefore, the initiator must know in advance the maximum number of hops the flood must propagate through. This poses no real practical limitation, as the maximum expected hop count can be estimated at the deployment time of the network. It follows that given a multi-hop network of k hops, the initiator must wait at least $(k+1)T_x$ seconds before transmitting the synchronization bit.

We show experimentally in Section 4.5 that a mean per-hop synchronization as low as 34 μs is achievable in a 2-hop network, while only taking on the order of tens of milliseconds to complete a Zippy flood.

4.4.3 Event Dissemination

We next take advantage of the per-hop time synchronization to rapidly propagate a fixed length packet through a multi-hop network. Since the OOK receiver provides bit-level granularity, we consider the bit-level dissemination of a small packet between single-hop participants.

The simplest method to propagate a bit through a multi-hop network is to transmit the bit as soon as it is received. This scheme is exemplified in Figure 4.8(a), where the initiator transmits bit b_0, participant node S_2

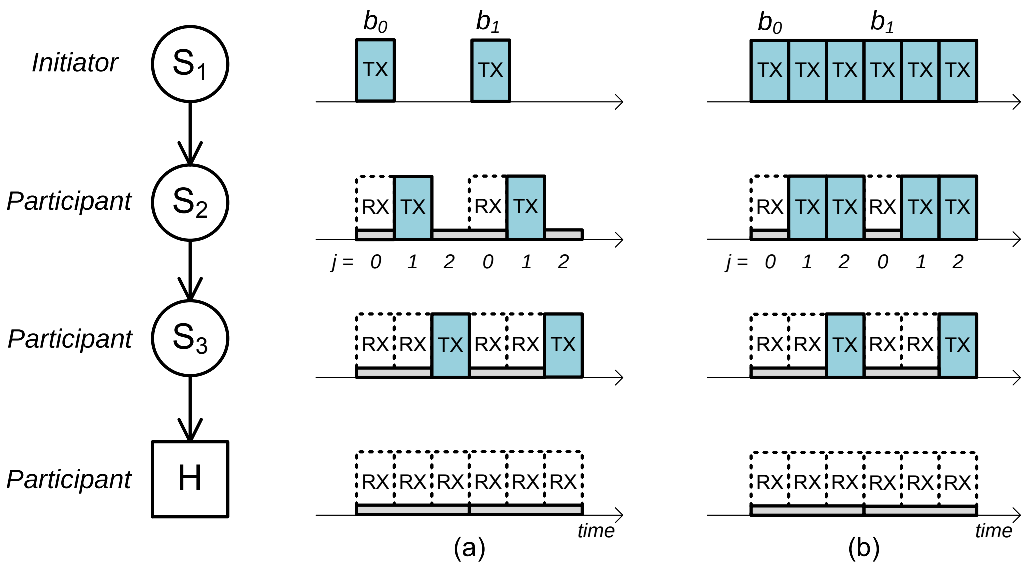


Figure 4.8: Bit-level dissemination schemes using $k = 3$ sub-bits (a) without, and (b) with repetition.

receives the bit and transmits it in the next available bit period. This bit-level relaying continues throughout the multi-hop topology. Under this scheme, the initiator must know in advance the maximum hop count k of the network in order to determine when it can transmit the next bit b_1.

We therefore impose a structure where each bit consists of k sub-bits, where k is the maximum hop count of the network. All participant nodes maintain a local counter j, which is initialized to zero. The counter represents the sub-bit number, i.e., sub-bit 0 to $k - 1$, currently decoded by the participant. The counter is incremented by the participant at the end of each symbol boundary according to the sequence given in (4.3).

$$j = (j + 1) \bmod k \quad \forall j \in [0, k - 1] \tag{4.3}$$

Once the sub-bit has been decoded, i.e., the *DATA* line has been sampled and is deemed as either a 0 or a 1, the participant makes a decision on how to propagate the sub-bit based on the value of the counter j. If a sub-bit is decoded as a 0, the participant continues to listen for the next sub-bit. However, if a sub-bit is decoded as a 1, the participant will transmit a 1 during the next sub-bit if its counter value satisfies $j < k - 1$. This behavior is illustrated in Figure 4.8(a), where node S_2 decodes a 1 during sub-bit $j = 0$, and therefore transmits a 1 in the next sub-bit. Similarly, node S_3 decodes a 1 during sub-bit $j = 1$, and transmits a 1 in the next sub-bit. This scheme of bit-level event dissemination ensures that all participants up to the k^{th} hop have an opportunity to receive each bit transmitted by the initiator.

The reliability of this scheme can be further improved by *(i)* incorporating a repetition code for the initiator, *(ii)* allowing participants to

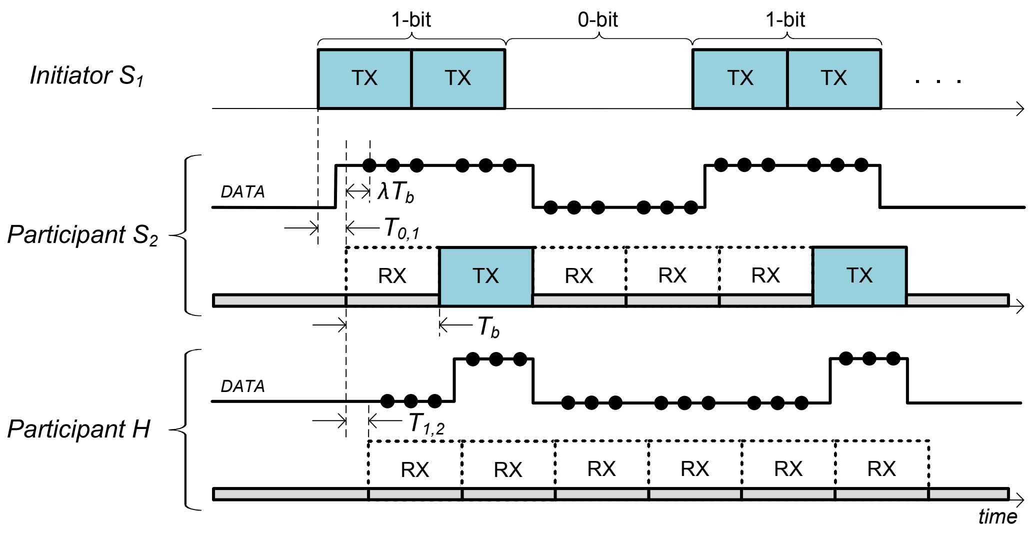


Figure 4.9: Bit-level dissemination employed by Zippy with $n = 3$ majority vote sub-bit decoding.

transmit during empty sub-bits, and *(iii)* performing a majority vote sub-bit decoding at all participants. We next detail these three techniques, which are all incorporated into Zippy.

As illustrated in Figure 4.8(b), the initiator of a Zippy flood employs a k-repetition code, whereby a single bit is transmitted repeated $k-1$ times. In order to take advantage of the transmission of the additional sub-bits from the initiator, each participant may propagate all remaining 1 sub-bits. That is, if a participant decodes the j^{th} sub-bit as a 1, the remaining $k-j-1$ sub-bits are then transmitted as 1 bits. To exemplify the dissemination scheme employed by Zippy, consider the example illustrated in Figure 4.9, where initiator S_1 transmits the first bit out of three, assuming a maximum hop count of $k = 2$. Participant S_2 will receive the first sub-bit as a 1, and then transmit during the remaining sub-bit, whereas participant H decodes the first sub-bit as a 0, and therefore refrains from transmission, and instead decodes the second sub-bit as a 1. This described scheme gives participants residing up to $k-1$ hops away from the initiator more than one chance at successfully decoding each bit.

Majority vote decoding, a scheme originally proposed in [Ree54], is performed at each participant by sampling the *DATA* line of its OOK receiver n times per bit period T_b, where n is a positive odd integer. The sub-bit is decoded as a 1 if the *DATA* line was sampled high at least $\left\lceil \frac{n}{2} \right\rceil$ times, otherwise, the sub-bit is decoded as 0. Figure 4.9 illustrates the timing diagram for majority vote sub-bit decoding, with each sample per sub-bit separated by λT_b, where $\lambda = \frac{1}{n+1}$.

In order for the majority vote sub-bit decoding to function as intended, the sampling of the *DATA* line must not extend past the sub-bit boundary.

This is prevented by ensuring the sub-bit period T_b is long enough, given the maximum per-hop synchronization delay $T_{i,i+1}$ experienced throughout the multi-hop network. This is analytically described by selecting the sub-bit period T_b such that the inequality given in (4.4) holds.

$$T_b > \frac{T_{i,i+1}}{\lambda} \quad \forall i \in [0, k-1] \tag{4.4}$$

It is evident that Zippy's event dissemination is inherently biased toward 1 bits, since a 0 bit requires all k sub-bits to be decoded as 0, while a 1 bit requires only one of the k sub-bits to be decoded as 1. While this asymmetry may lead to packet corruption due to erroneous 1 bit dissemination, Zippy utilizes the tight per-hop synchronization for aligning sub-bit boundaries, and majority sub-bit decoding for improved packet reception as evaluated in Section 4.4.4.

An important property of Zippy's event dissemination scheme is that the end-to-end latency of a flood at each node differs only by the small per-hop synchronization delays $T_{i,i+1}$. Specifically, the minimum end-to-end latency $L_{0,h}$ from the beginning of the initiator transmission until the participant located h hops away decodes the N_{data} bit packet, is given by:

$$L_{0,h} \geq T_{preamble} + (k+1)T_x + \sum_{i=0}^{h-1} T_{i,i+1} + (kN_{data} + 1)T_b \tag{4.5}$$

We can therefore conclude that the theoretical minimum end-to-end latency $L_{0,h}$ increases linearly with the maximum hop count k, and with the number of packet bits N_{data}.

4.4.4 Carrier Frequency Randomization

We have so far only considered the operation of Zippy in a network of nodes arranged in a line topology. We now extend its design to more realistic wireless networks where each hop may contain several participating nodes.

The main challenge in extending Zippy into a dense network is interference. For example, when two nodes simultaneously transmit an OOK-modulated 1 bit, the corresponding sinusoidal signals may constructively or destructively interfere at the antenna of the OOK receiver residing at the next hop. If the interference is constructive, then the receiver will decode the bit successfully without any adverse side effects. However, if the interference is destructive, the received signal level may be too low to be detected.

During a Zippy flood, destructive interference is highly undesirable as all of its components may be affected. For example, destructive

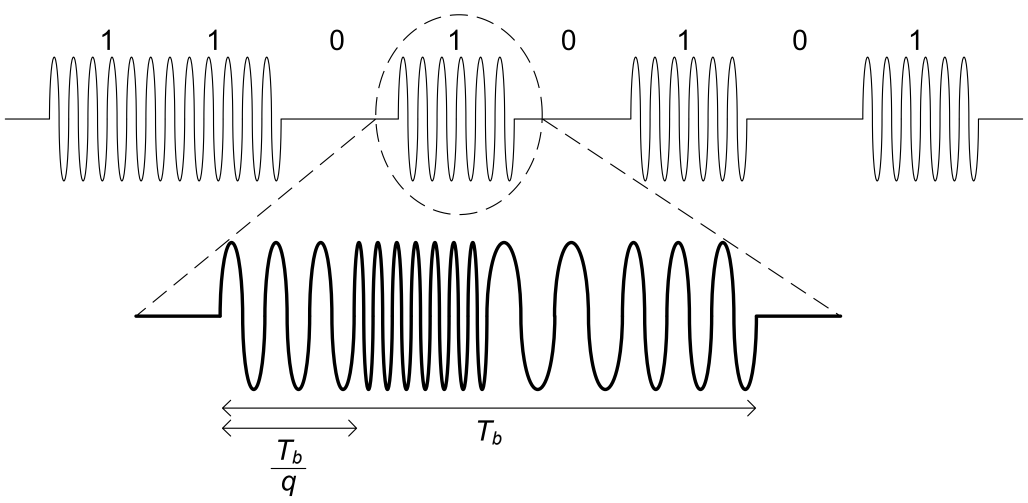


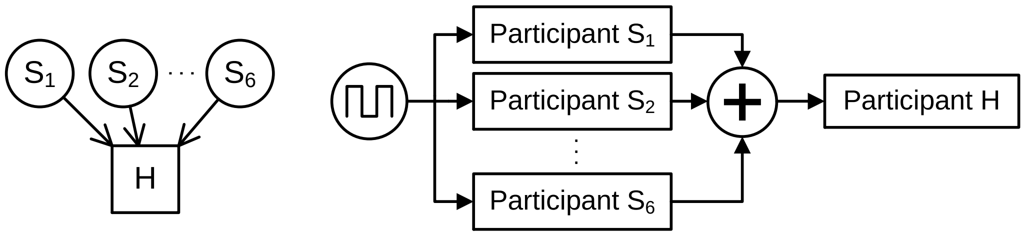
Figure 4.10: An example OOK transmission using carrier frequency randomization parameterized with $q = 4$.

interference may result in a missed wake-up preamble reception, erroneous per-hop synchronization due to the late reception of the synchronization bit, or packet corruption due to erroneous decoding of sub-bits.

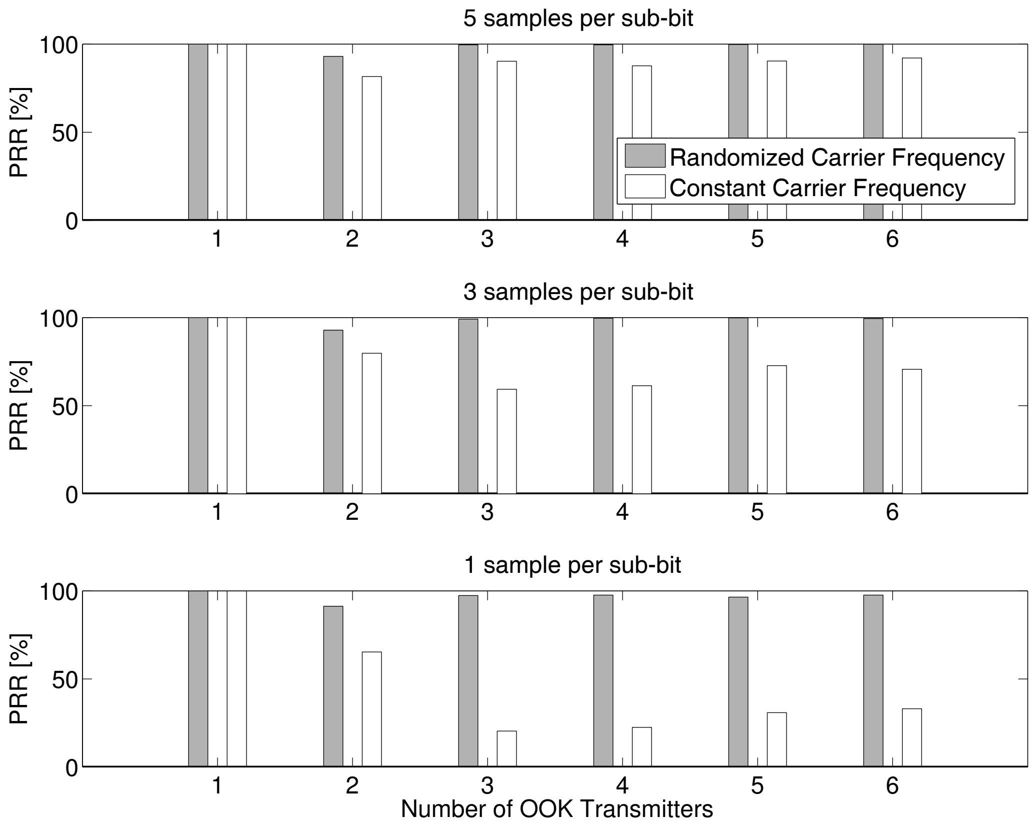
One solution to this problem is to avoid the interference by employing a collision avoidance mechanism, e.g., clear channel assessment with random back-off. However, there are two severe problems that arise, making such mechanisms infeasible for Zippy. Firstly, the use of a random back-off would introduce long delays between the relaying of each wake-up preamble. This will make it impossible to tightly bound the delay T_x, which is used by the initiator and all participants to correctly sequence the synchronization bit. Secondly, the induced delays of relaying the synchronization bit will drastically extend the per-hop synchronization bounds $T_{i,i+1}$, leading to an excessively long OOK bit period T_b in order to satisfy inequality (4.4) presented in Section 4.4.3.

Therefore, rather than trying to avoid collisions, Zippy embraces overlapping transmissions, but mitigates their adverse effects by randomizing the carrier frequency used to generate OOK 1 sub-bits. Instead of transmitting each OOK 1 sub-bit using a single carrier frequency of duration T_b, each sub-bit is transmitted using a random sequence of $q > 1$ carrier frequencies, as exemplified in Figure 4.10. Since each node randomly selects a different set of carrier frequencies per sub-bit, the probability of destructive interference during the entire sub-bit reduces significantly.

In particular, each node selects a carrier frequency $f = f_c + \beta\Delta$, at least q times per sub-bit period T_b, where f_c is the center carrier frequency, β is a discrete random variable uniformly distributed over a set of integers of size $M \geq 2$, and Δ is a minimum frequency offset.



(a)



(b)

Figure 4.11: (a) Logical and physical experimental setup, and (b) results of evaluating carrier frequency randomization and majority vote sub-bit decoding.

The parametrization of q, M, and Δ are dependent on the hardware of the OOK receiver, OOK transmitter, and the microcontroller used to control the two devices. Based on the developed prototype detailed in Section 4.5.1, together with extensive practical experiments, we support carrier frequency randomization with $q = 8$ random frequencies per sub-bit, with $M = 4$ random frequencies centered about $f_c = 446.8\,\text{MHz}$, with each having a minimum offset of $\Delta = 135.4\,\text{kHz}$.

We next experimentally evaluate the effectiveness of carrier frequency randomization based on the aforementioned parameterization using the cabled setup as illustrated in Figure 4.11(a). Using a 6-way RF combiner, we combine the simultaneous transmissions of up to six participants, with the resultant signal cabled to an independent participant for decoding using its OOK receiver. An external signal generator is used to trigger participants $S_1, S_2, \ldots, S_6$ to transmit a 32-bit packet containing 24-bits of randomly generated payload and an 8-bit cyclic redundancy check. All participants are configured with the same seed for the pseudo-random number generator, ensuring that each packet is identical during each simultaneous transmission. However, each participant produces a randomized carrier frequency using its own unique seed. The packet reception rate (PRR) is evaluated at participant H using the cyclic redundancy check for with and without carrier frequency randomization, and with and without majority vote sub-bit decoding. A total of a 1000 packets are generated for each test configuration. The transmission power of all participants is fixed at −30 dBm. Shielded SMA cables interconnect all RF ports in order to remove non-deterministic propagation effects.

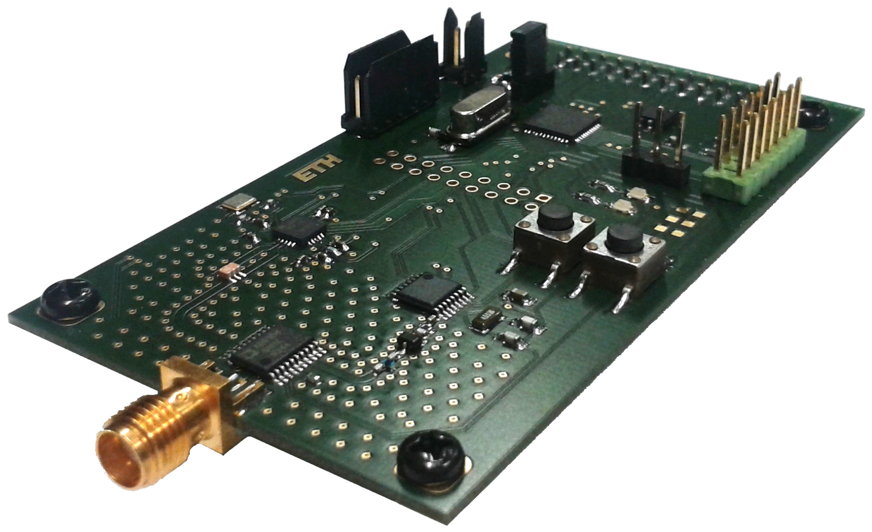
The results of the experiment are illustrated in Figure 4.11(b). As expected, a 100% PRR is achieved for a single transmitter, irrespective of the carrier frequency and majority decoding configuration. As the number of simultaneous transmitters is increased, the PRR drastically reduces when using a constant carrier frequency. This effect is slightly mitigated by increasing the number of samples per sub-bit from 1 to 5. In the case of only two simultaneous transmitters, we observed a slight reduction of PRR when using a randomized carrier frequency. Since the selection of carrier frequency is a random process, we would expect the occasional bit error due to destructive interference. Nevertheless, the PRR remains higher than that achieved using a constant carrier frequency. In summary, carrier frequency randomization achieves almost 100% PRR with up to six simultaneous transmitters, thus making it a robust mechanism for Zippy to operate in dense networks.

4.5 Experimental Evaluation

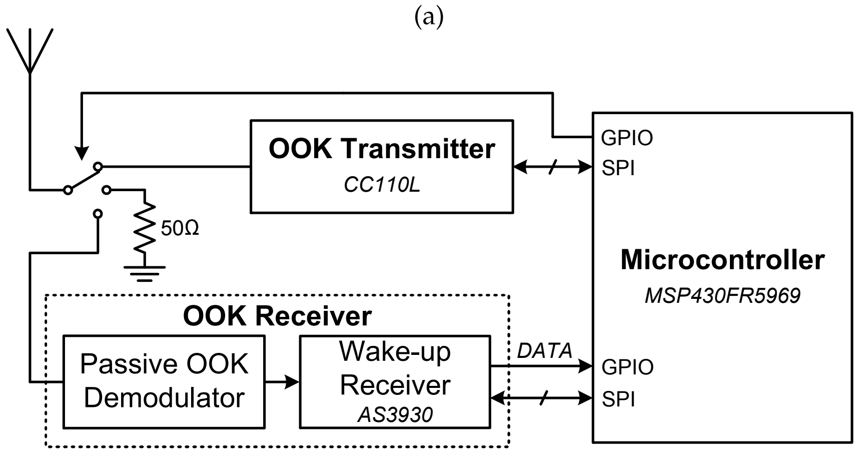
In this section, we present the design of a custom wireless embedded platform for the evaluation of Zippy. We first present the platform design, evaluate the sensitivity and erroneous wake-up rate of its integrated ultra-low power OOK receiver, and measure the platform's power dissipation. We then evaluate the performance of Zippy in a cabled laboratory setting and in a wireless indoor testbed.

4.5.1 Prototype Wireless Embedded Platform

A prototype wireless embedded platform supporting Zippy is shown in Figure 4.12(a), with its block diagram depicted in Figure 4.12(b). The



(a)



(b)

Figure 4.12: Prototype (a) and block diagram (b) of a wireless embedded platform supporting Zippy.

platform consists of a 16-bit MSP430FR5969 microcontroller running at 16 MHz interfaced to a CC110L transceiver, and an ultra-low power OOK receiver based on the AS3930 receiver. The antenna of the platform is connected to an ADG904 RF switch, which is controlled by the microcontroller. The antenna path can be fed into the OOK transmitter, the OOK receiver, or a 50 Ω resistor. The components of the platform were selected based on their ultra-low quiescent current drain or ultra-low current drain during sleep mode.

OOK Receiver. The receiver is an adaptation of [GSKR10], where the impedance matching circuitry has been tuned to the 434 MHz ISM frequency band. The receiver consists of a passive OOK demodulator coupled with a commercially available AS3930 ASK receiver. We use the *WAKE* line from the AS3930 to indicate the reception of a wake-up

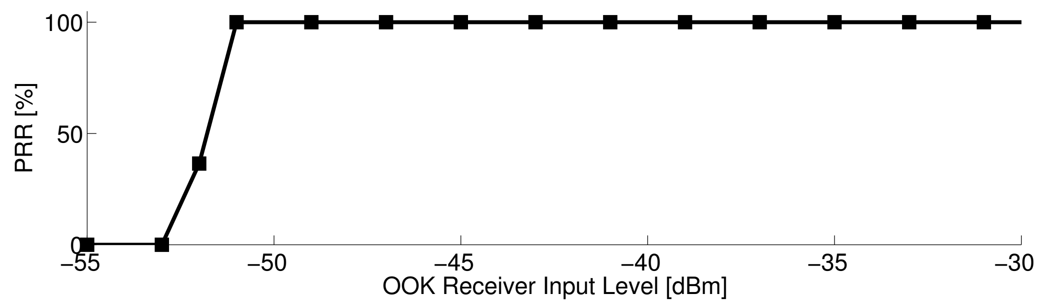


Figure 4.13: Packet reception rate of the OOK receiver as the input signal level is varied using an attenuator. A sensitivity of approximately −52 dBm was measured.

preamble, while the *DATA* line is used for detecting the synchronization bit and decoding packets. The OOK receiver supports a maximum data rate of 8.192 kbps, and features an ultra-low current drain of 2.7 μA measured at 3.0 V.

OOK Transmitter. In order to design a flexible wireless sensor platform for future research opportunities, we incorporated a multi-purpose wireless transceiver instead of a dedicated OOK transmitter module. Carrier frequency randomization is supported on the CC110L transceiver by configuring it for transmission using FSK-4 modulation. An OOK signal is produced by continuously generating random FSK-4 symbols, and using the antenna switch to generate either a 1 or a 0 sub-bit, as required. An OOK 1 sub-bit is produced by switching the antenna port to the CC110L transceiver for a period of T_b, while an OOK 0 sub-bit is produced by switching the antenna port to the 50 Ω resistor for a period of T_b. A software-based pseudo-random number generator [Mar94] is used to randomly select each FSK symbol from a uniform distribution, thus producing the desired randomization of the carrier frequency.

4.5.1.1 OOK Receiver Sensitivity

The sensitivity of a wireless receiver is defined as the minimum input signal power that supports successful decoding. While the receiver sensitivity has an influence on the overall reception range of the receiver, care must be taken not to infer reception range only from receiver sensitivity, as various techniques can be used to improve the RF link budget, e.g., increasing the power of the transmitted signal and the use of high-gain antennas.

We measured the sensitivity of the OOK receiver using the experimental setup described in Section 4.4.1, and illustrated in Figure 4.5. By varying the attenuation between OOK transmitter and receiver, the sensitivity is determined by the minimum input signal level upon which

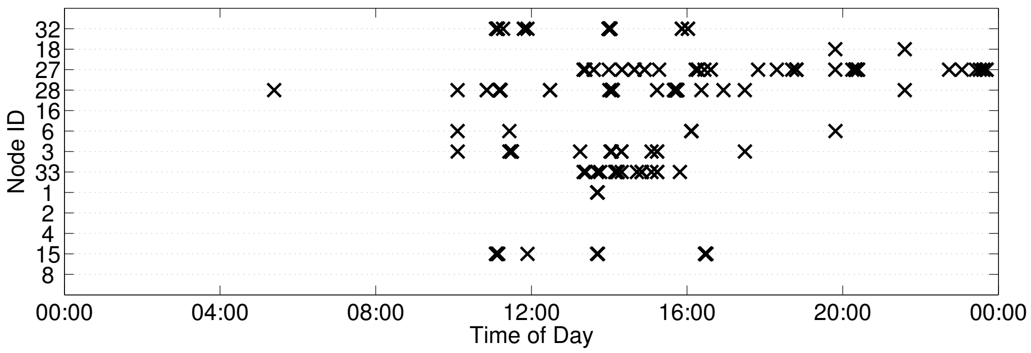


Figure 4.14: Erroneous wake-ups observed during a 24 hour experiment using Zippy nodes deployed in the FlockLab indoor testbed.

the *DATA* line no longer asserts in the presence of an OOK 1 sub-bit transmission. As shown in Figure 4.13, the receiver sensitivity of the OOK receiver is approximately −52 dBm. This sensitivity level is comparable to existing prototypes exhibiting a similar receiver design [GR12].

It is important to clarify that despite the apparent low sensitivity compared to high-complexity receivers, practical experiments have demonstrated ranges up to 15 meters non-line-of-sight in an office hallway, and up to 30 meters line-of-sight in an outdoor sports field. These experiments were carried out using quarter-wavelength omnidirectional monopole antennas with a square ground plane having a surface area of 25 cm^2, while being fixed 1.5 meters above the ground.

4.5.1.2 Erroneous Wake-up Rate

Due to the low-complexity design of the OOK receiver, it is susceptible to external interference. In particular, we are interested in how often an erroneous wake-up occurs, i.e., when the *DATA* line asserts without an OOK preamble being transmitted. The OOK receiver used in this work operates using sub-carrier modulation [ODC+13a], where a 434 MHz carrier is used to generate the envelope of a 125 kHz carrier. While this design achieves ultra-low power dissipation, it is susceptible to strong noise sources operating near the 125 kHz frequency, e.g., switch-mode power supplies, and wireless devices operating in the 434 MHz ISM band.

In order to quantify the erroneous wake-up rate of our prototype, we deployed 13 Zippy nodes into an indoor office environment, as detailed in Section 4.5.3. During a period of 24 hours, we recorded each time an erroneous wake-up occurred, i.e., the rising edge of the *DATA* line without a controlled transmission of a wake-up preamble. The results of the experiment are shown in Figure 4.14, where a cross indicates an erroneous wake-up. In summary, four nodes did not exhibit any erroneous wake-ups, while node 28 experienced the maximum of 3166

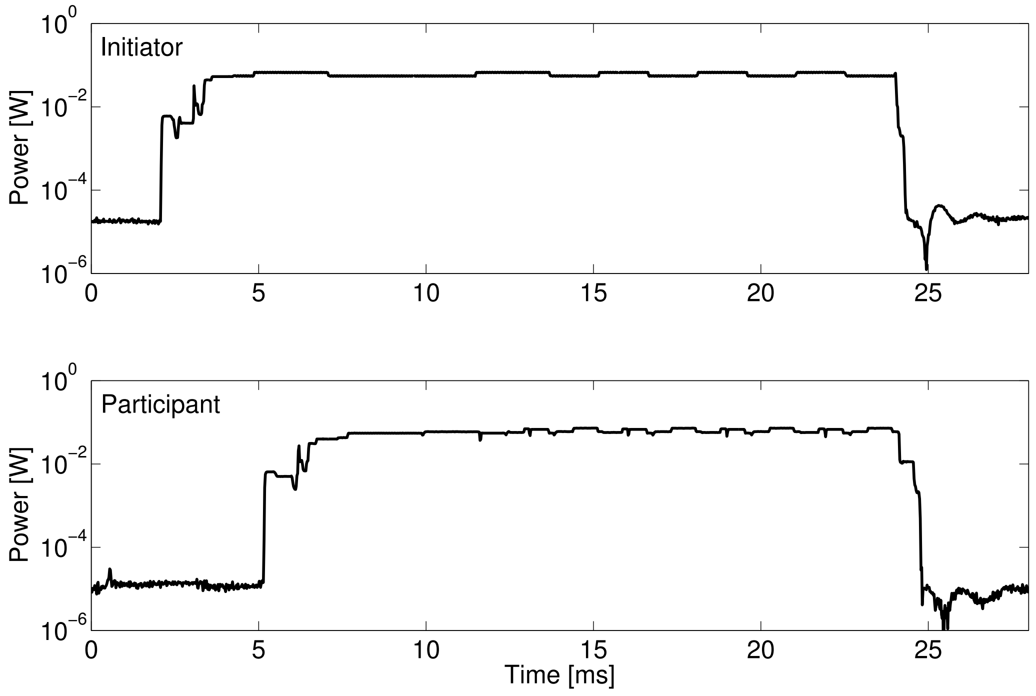


Figure 4.15: Power profile of initiator and participant nodes during a Zippy flood. The nodes dissipate approximately 70 mW during the Zippy flood, and 9.6 μW during periods of inactivity.

erroneous wake-ups. With the exception of node 27, the erroneous wake-ups were predominantly recorded during office hours. Due to the limited observability of the input signal level at each node, it is difficult to determine the root cause of the observed erroneous wake-ups. Nevertheless, we present a novel scheme for mitigating erroneous wake-ups in Chapter 5.

4.5.1.3 Power Dissipation

The power dissipation of an initiator and participant node was measured using an Agilent N6705A DC power analyzer. The two platforms were supplied with 3.0 V, while having their on-board low-dropout regulators bypassed. The collected power profiles are depicted in Figure 4.15.

The power profiles begin with both nodes in a deep sleep listening state, whereby the microcontroller resides in a low-power sleep mode, i.e., LPM4, the OOK transmitter is in standby mode, the antenna switch is active, and the OOK receiver is active. The power dissipation of the initiator and a participant nodes during this state was measured as 9.6 μW, which is low enough to facilitate multi-year operation using low-capacity coin cell batteries. At approximately 2 ms into the power profile, the initiator is triggered to start a Zippy flood parameterized with

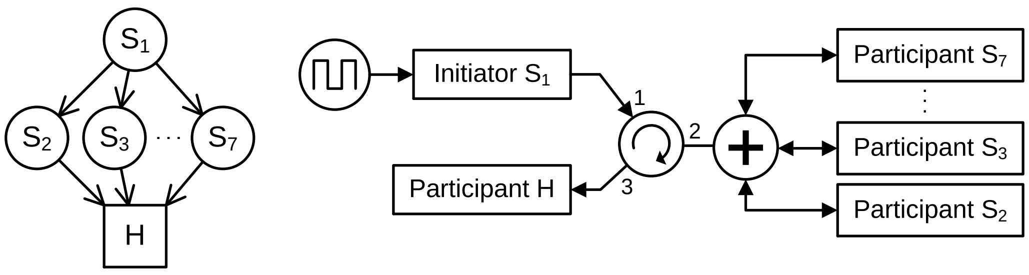


Figure 4.16: Logical (left) and physical (right) setup for the cabled multi-hop experiments.

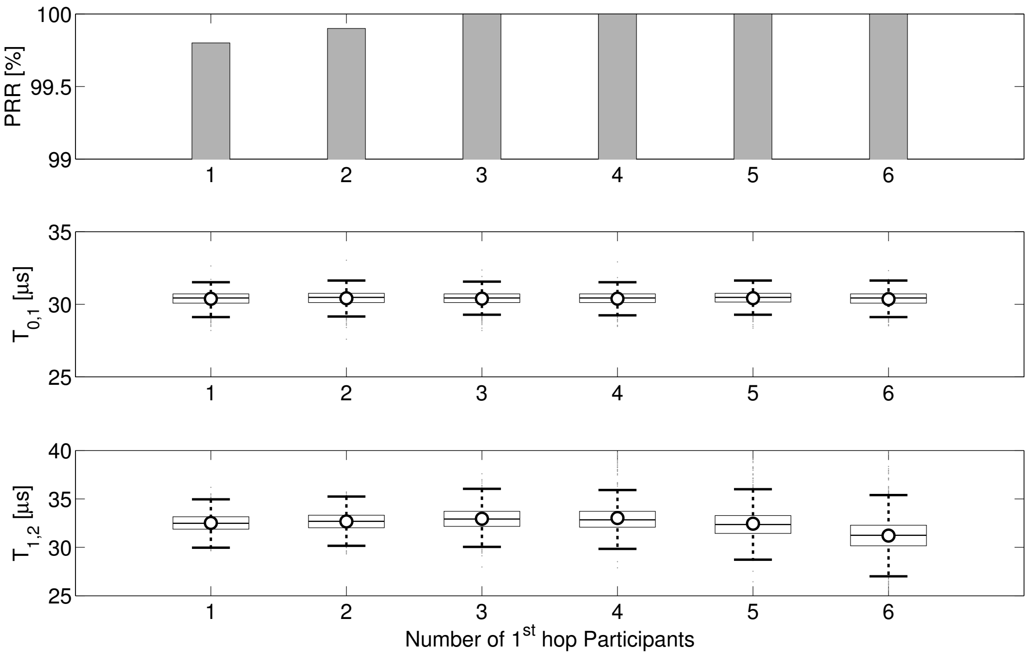


Figure 4.17: Packet reception rate and per-hop time synchronization of the cabled multi-hop network.

a maximum of $k = 2$ hops. The initiator begins to transmit the wake-up preamble, followed by the synchronization bit, and an 8-bit packet containing the bit sequence 0x55. The power dissipation during all OOK transmissions is approximately 70 mW, which concurs with the datasheet of the transmitter.

4.5.2 Cabled Multi-hop Experiments

We first evaluate Zippy in cabled laboratory experiments before extending the evaluation to a wireless indoor testbed. The motivation for cabled experiments is two-fold. Firstly, to provide a baseline timing analysis of the per-hop synchronization achieved by Zippy, and secondly, to evaluate the robustness of Zippy in dense networks that would otherwise be infeasible to replicate in an indoor testbed.

Setup. As illustrated in Figure 4.16, a cabled 2-hop network is used to evaluate the timing and robustness of Zippy using a circulator. A circulator is a passive 3-port device that passes an RF signal between ports 1 and 2, but isolates the signal from port 3. Similarly, a signal may pass between port 2 and 3, but is isolated from port 1. Connecting a 6-port RF combiner at port 2 of the circulator enables the construction of sparse and dense 2-hop network topologies. A logic analyzer is connected to initiator S_1, and participants S_2 and H for the measurement of the per-hop time synchronization and evaluation of packet reception rate.

The initiator S_1 starts the transmission of each Zippy flood according to the frequency of a square-wave signal generator. Each flood consists of a 32-bit packet containing 24 randomly generated bits and an 8-bit CRC. Zippy is configured with $k = 3$ maximum hops, majority decoding with $n = 3$ samples per bit, and with carrier frequency randomization as described in Section 4.4.4. The transmission power of all nodes is fixed at -30 dBm. The experiment begins with a single 1-hop participant, S_2, with all other combiner ports terminated with $50\,\Omega$ SMA terminators. After the transmission of 1000 Zippy floods by initiator S_1, an additional 1-hop participant, up to a maximum of six, is connected to the cabled multi-hop network before restarting the experiment.

Results. The results of the cabled laboratory experiment are shown in Figure 4.17. The robustness of Zippy is determined by the packet reception rate as measured at participant H. As more 1-hop participants are added, it is vital that the packet reception rate does not decrease. As we can see from Figure 4.17, the packet reception rate remains at approximately 100% with up to six first-hop participants. This remarkable behavior is primarily attributed to the use of carrier frequency randomization. As discussed in Section 4.4.4, the use of majority vote decoding slightly increases the packet reception rate under overlapping transmissions. This experimental result demonstrates an improvement on the packet reception rate achievable during concurrent OOK transmissions compared to using high-complexity receivers as presented in [WLS14], where simulations show that approximately 10% of all packets are corrupt when there is more than one concurrent transmission.

The per-hop time synchronization is determined by measuring the $T_{i,i+1}$ delay, as introduced in Section 4.4.2. The first-hop synchronization between the initiator S_1 and participant S_2 is given by $T_{0,1}$, while the second-hop synchronization between participants S_2 and H is given by $T_{1,2}$. As shown in Figure 4.17, the mean of the first-hop time synchronization is approximately $31\,\mu$s, and exhibits a very narrow

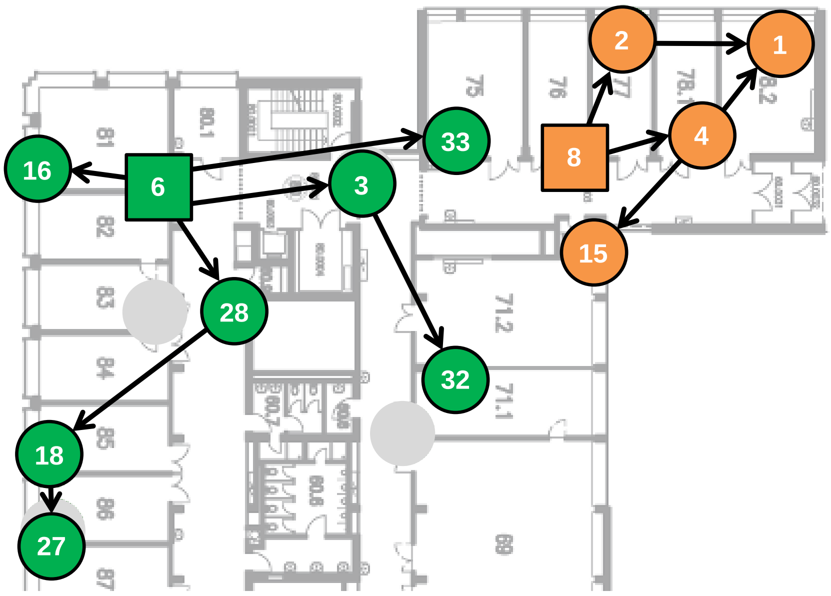


Figure 4.18: FlockLab indoor testbed deployment map for evaluating Zippy, featuring large (left) and small (right) network topologies.

distribution, irrespective of the number of additional participants. This result is not surprising, since the reception of the synchronization bit at S_2 is not affected by the reception of the synchronization bit by all other participants residing at the same hop. However, the mean of the second-hop time synchronization is slightly increased to approximately 33 μs, and has a wider distribution due to the colliding synchronization bit transmissions. As the number of 1-hop participants increases, the percentiles of the time distribution increase accordingly, however, the mean remains below 35 μs. This demonstrates that under ideal conditions, Zippy provides fine-grained per-hop time synchronization, while also exhibiting robustness in sparse and dense network topologies due to carrier frequency randomization.

4.5.3 Testbed Multi-hop Experiments

We next evaluate Zippy's performance in the FlockLab indoor testbed.

Setup. We deployed a total of 13 prototype wireless embedded platforms supporting Zippy into the FlockLab [LFZ+13b] indoor testbed, which is configured with fine-grained tracing capabilities [LMD+15]. The location of each deployed node is depicted in Figure 4.18. We configured the nodes into two independent networks, a 2-hop network with a topology having more than one participant at the first hop, and a 3-hop network offering substantial spatial coverage. All nodes were configured with carrier frequency randomization, as detailed in Section 4.4.4. Nodes 8

Table 4.1: Zippy configuration used throughout all indoor testbed experiments.

Parameter	**Description**
$k = 2$ and 3 hops	Maximum hop count
$D = \frac{1}{T_b} = 1.364$ kbps	OOK receiver data rate
$N_{data} = 8$ and 16 bits	Packet length
$n = 3$ samples per sub-bit	Majority vote sub-bit decoding
$T_{preamble} = 1.4\,\text{ms}$	Preamble duration
$T_x = 1.25\,\text{ms}$	Participant wait delay

and 6 were configured as the initiators for the small and large networks, respectively. All experiments were performed with 8-bit and 16-bit randomly generated packets. As discussed in Section 4.5.4, support for longer packets with Zippy is indeed feasible.

The experimental evaluation is based on the following four metrics: *(i) wake-up reception rate (WRR)* is the ratio of the number of nodes that wake-up compared to the number of Zippy floods initiated, *(ii) packet reception rate (PRR)* is the ratio of the number of correctly received packets compared to the number of Zippy floods initiated, *(iii) transmission time* is the duration each node had their OOK transmitter active per Zippy flood, *(iv) end-to-end latency* is the elapsed time between the start of the Zippy flood at the initiator until the end of the flood at each participant, and *(v) per-hop synchronization* is the mean time delay between the nodes' nearest neighbor during all Zippy floods. We compute all metrics based on 500 Zippy floods using a static configuration, as listed in Table 4.1.

Results. The results of the testbed experiments are shown in Figure 4.19, and are summarized as follows:

- We first observe a wake-up reception rate (WRR) of 100% for all nodes deployed in the testbed. This is a remarkable result, given the challenging RF propagation environment imposed by the testbed: nodes are separated by physical obstacles including thick concrete walls reinforced with steel, and metal piping affixed to the ceiling for plumbing, heating and ventilation. No erroneous wake-ups were observed during the experiments.

- The packet reception rate (PRR) of the nodes residing in the large network range from 1.8% (node 27) up to 100% (nodes 16, 3, and 33), while all nodes residing in the small network exhibited a PPR exceeding 94.6%. The higher PRR in the small network is attributed to the improved link quality between nodes, which exhibit shorter link distances and less physical obstruction compared to the large network topology.

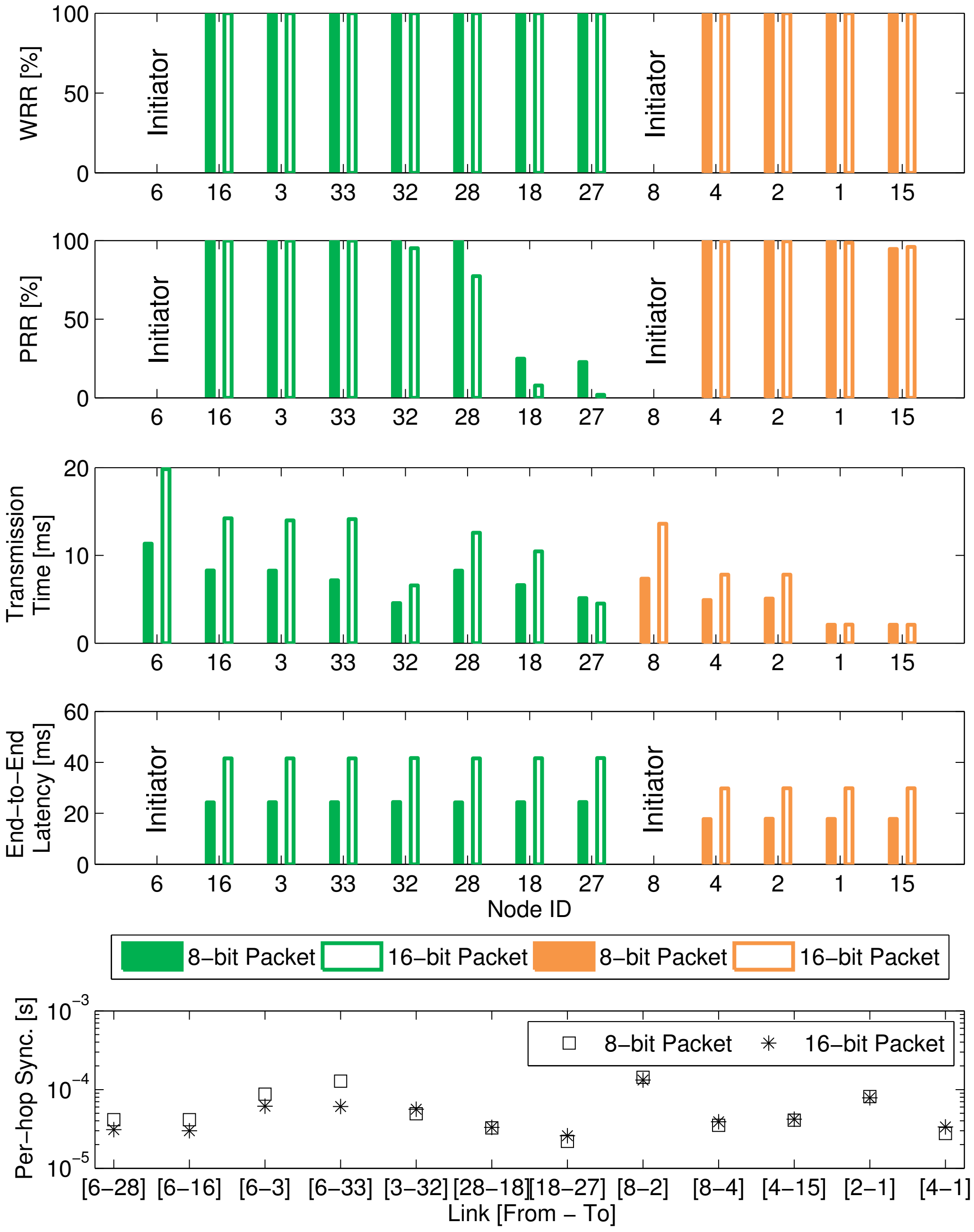


Figure 4.19: Results from indoor testbed experiments using Zippy in small and large network topologies.

- The transmission times are approximately equal for all nodes having the same hop count. This is due to the way in which Zippy disseminates each sub-bit of the packet. In addition to the wake-up preamble and synchronization bit transmissions, each initiator (nodes 6 and 8) transmit all packet bits using a repetition code, as detailed in Section 4.4.3. As the hop count increases, the number of sub-bits to be propagated decreases, resulting in a decrease in transmission time per hop. For example, in the small network, node 8 initiates the Zippy flood and takes 7.3 ms to transmit the wake-up preamble, synchronization bit, and an 8-bit packet, while nodes 2 and 4, residing at the first hop, experience a transmission time of approximately 5.0 ms for the wake-up preamble, synchronization bit and only half of the packet sub-bits. Finally, the leaf nodes, i.e., nodes 1 and 15, require a transmission time of 2.0 ms, as they transmit only the wake-up preamble and synchronization bit. A maximum of 19.8 ms was observed by the initiator of the large network for the transmission of a 16-bit packet.

- The end-to-end latency is nearly constant for each network topology and evaluated packet length. Due to the way in which Zippy propagates sub-bits through the network, all participant nodes complete a flood at approximately the same time, with small differences attributed to the per-hop time synchronization to their neighbors. The small and large networks exhibit average end-to-end latencies of 17.8 ms and 24.4 ms for the 8-bit packet, and 29.8 ms and 41.6 ms for the 16-bit packet, respectively.

- The mean per-hop synchronization ranges from 21.9 μs between nodes 18 and 27, and 143.5 μs between nodes 8 and 2. Apart from the link between nodes 6 and 33, there are only slight differences between the 8-bit and 16-bit packet experiments. This is to be expected as the per-hop time synchronization is dependent on the reception of the synchronization bit and not on the length of the packet. It is important to highlight that the minimum per-hop synchronization observed is less than what was measured in the cabled experiments in Section 4.5.2. This is indeed feasible, as all links in the testbed are not perfectly isolated. For example, node 27 may on occasion receive the synchronization bit transmitted from node 28, while the event dissemination is decoded from node 18.

Comparative Analysis. We next compare the energy efficiency of Zippy in flooding on-demand events compared to state-of-the-art flooding protocols, such as Glossy [FZTS11]. We consider a static network topology with a maximum of $k = 3$ hops and a packet length of 16-bits. As we have shown in the testbed experiments, Zippy can disseminate an event through a 3-hop network with an end-to-end latency of approximately 41.6 ms. If instead Glossy was used to disseminate on-demand events with a reporting latency comparable to Zippy, all nodes must perform periodic Glossy floods with a periodicity of at most 41.6 ms. Assuming the Glossy parameterization presented in [Zim15] for the TelosB and CC2420 radio with a maximum number of transmissions $N = 2$, this corresponds to maximum transmission time of 1.9 ms per flood every 41.6 ms. In contrast, the Zippy prototype only needs a maximum transmission time of 19.8 ms once per event. Despite the radio-on time depending greatly on the specific node hardware and the protocol software implementation, we can conclude that Zippy will consume less energy than periodic Glossy floods provided the average event inter-arrival time is greater than approximately 434 ms.

4.5.4 Protocol Limitations and Challenges

We next discuss limitations and challenges associated with the operation, implementation and evaluation of Zippy.

Erroneous Wake-ups. As discussed in Section 4.5.1.2, the always-on low-complexity OOK receiver is susceptible to erroneous wake-ups. Zippy is designed such that a wake-up preamble is always transmitted once a wake-up is detected. Therefore in the worst-case scenario, if an erroneous wake-up is detected at one node, all nodes in the network will erroneously participate in a Zippy flood. After the wake-up preamble has been erroneously relayed, two scenarios are possible, namely, *(i)* a synchronization bit is detected and an erroneous packet is decoded and delivered to a higher-layer application, or *(ii)* a synchronization bit is not detected causing a protocol timeout after the expiration of a timer. Erroneous packets can be filtered at higher-layers using forward error correction, however, in either case, an erroneous wake-up results in the waste of energy resources. In order to mitigate this energy waste, we propose a novel technique to mitigate erroneous wake-ups in Chapter 5.

Erroneous Synchronization. If a participant node receives the synchronization bit too early due to a erroneous wake-up, or too late due to poor RF propagation, subsequent transmissions may corrupt the reception of all other participants within its 1-hop neighborhood. Extensive tests suggest an adequate parameterization of the wake-up preamble duration $T_{preamble}$ and participant wait delay T_x reduce the occurrence of this erroneous behavior.

Data Rate Reduction. The data rate of the OOK receiver had to be reduced to 1.364 kbps for the implementation of Zippy due to instability of the OOK demodulator output. We suspect that this is due to the dynamic operation of the AS3930's integrated AGC. Since the design of the AGC is not in the public domain, it is difficult to identify the root cause of this behavior. However, in principle, there are no limitations in operating Zippy at higher data rates using an appropriately designed AGC.

Packet Length. The packet length supported by Zippy is primarily dependent on the quality of the RF links between nodes. As exemplified in Section 4.5.3, nodes with poor RF connectivity, e.g., nodes 18 and 27 in the large network, will experience higher bit errors, resulting in a lower PRR as the packet length is increased. However, when nodes have good RF connectivity, longer packet lengths may be supported with a high PRR. For example, in the small testbed network, Zippy has been shown to support a packet length of 64-bits with an average PRR of 96.7%.

Network Scalability. As shown in the experiments presented in Section 4.5.3, Zippy's end-to-end latency is approximately constant for a given network configuration, and increases linearly with respect to the maximum hop count k, and the number of packet bits N_{data}. Furthermore, the transmission time of the initiator is determined by the maximum hop count k, while the transmission time of participants decreases linearly with respect to its hop count. It is anticipated that the advantageous properties of carrier frequency randomization, as validated in Section 4.4.4, will also apply in large network deployments with appropriate parameterization.

Antenna Ground Plane. The ground plane and placement of the antenna impacts the range of the low-complexity OOK receiver. Since each FlockLab observer has a defined physical footprint, the size of the antenna ground plane is therefore limited. Furthermore, by affixing the antennas close to thick reinforced concrete walls, as is the case for all indoor FlockLab observers, the antenna radiation pattern of each node is altered. These combined effects reduced the operation range of some links within the indoor testbed deployment, necessitating the installation of larger antenna ground planes at selected FlockLab observers.

4.6 Related Work

Asynchronous network wake-up was first proposed as part of the PTW [YV04] protocol, where each node immediately transmits a preamble as soon as it is woken up by the reception of a preamble. The relaying of preambles has been recently extended in FLOOD-WUP [PSTT14] by taking advantage of an addressable wake-up mode supported by modern ultra-low power receivers. Zippy builds on their work by tackling the problem of nodes simultaneously transmitting. Utilizing carrier frequency randomization, as introduced in this chapter, provides a robust, low-latency, and topology-independent asynchronous network wake-up. In addition, Zippy provides tight per-hop synchronization and low-latency event dissemination.

Several protocols employing ultra-low power receivers have been proposed in the literature, for example E2RMAC [JBA07], WUR-MAC [MD09], and RTWAC [APM09], which provide single-hop packet transfer using asynchronous rendezvous. While these protocols achieve improved end-to-end latency and energy efficiency compared to synchronous and pseudo-asynchronous protocols, they do not support asynchronous multi-hop event dissemination as demonstrated by ZIPPY. Furthermore, the aforementioned protocols have only been evaluated using simulation, in contrast to the indoor testbed deployment of ZIPPY.

Asynchronous rendezvous in wireless sensor networks have been proposed using RFID active [JRO10] and passive [BDH10] tags. While simulations of RFID-based multi-hop communication indicate promising energy efficiencies [JRO08], the high-power dissipation of the RFID reader and limited operational range severely prohibit the realization of energy-efficient multi-hop dissemination.

Ambient backscatter [LPT+13] is a novel communication paradigm, however it is dependent on the availability of high-power carrier signals generated by fixed infrastructure. In this work, we introduced an infrastructure-less approach to asynchronous rendezvous with ZIPPY, and demonstrated its operation in an indoor testbed. Despite the challenging RF propagation conditions, testbed experiments verify multi-hop connectivity with per-hop links of up to 10 meters.

Dual-radio wireless sensor platforms incorporating a high-powered transceiver and an ultra-low power receiver have been proposed in [BFE+07, PSMJ13, ODCP13, GSR14]. Although these prototype platforms support asynchronous rendezvous and single-hop event dissemination, they do not support carrier frequency randomization, and have not been experimentally evaluated in a testbed. We have shown experimentally using a custom-built prototype wireless sensor platform that carrier frequency randomization is a prerequisite for robust multi-hop event dissemination using low-complexity OOK receivers.

4.7 Summary

In this chapter we focus on the problem of disseminating rare events through a multi-hop network with low-latency and without wasting significant energy on unnecessary periodic communication. Instead of employing a periodic communication scheme using commodity RF-based receivers, we take advantage of low-complexity receivers that dissipate several orders of magnitude less power, while facilitating the on-demand flooding of rare events. We present ZIPPY, an asynchronous protocol for waking-up a multi-hop network of nodes on-demand, synchronizing each node to their nearest neighbor, before rapidly disseminating a small event packet to a remote host. We detail the design, analysis and prototype implementation of Zippy and experimentally evaluate its performance in a laboratory setting and in an indoor testbed. The results of the evaluation show that Zippy supports on-demand flooding with an end-to-end latency of 24.4 ms for an 8-bit packet through a 3-hop network, a per-hop synchronization as low as 21.9 μs, and a power dissipation of 9.6 μW during periods of inactivity.

5 Low Latency and Energy-efficient Event-triggered Wireless Communication

The design of state-of-the-art protocols for wireless multi-hop networks are built upon each node in the network periodically communicating with its one-hop neighbors. While this approach has been shown to be very efficient for periodically disseminating information through a network of resource constrained nodes, the performance of these protocols is severely limited when information is to be disseminated on-demand, i.e., upon the detection of an event. This severe limitation stems from the coupling between the worst-case latency of the dissemination as determined by the time between epochs of periodic communication, and the energy consumed by performing periodic communication during periods of time when no events of interest are detected.

In this chapter, we extend the state-of-the-art with Blitz, a novel communication architecture that decouples this fundamental trade-off to realize both low latency and energy-efficient on-demand dissemination of information, i.e., an event and its associated data, through a multi-hop network of resource-constrained nodes. Blitz combines two orthogonal communication primitives to achieve event-triggered wireless communications. Blitz employs ultra-low power wake-up receivers to asynchronously wake-up the multi-hop network, which is then followed by the synchronous dissemination of the event and its associated data using constructive interference. In order to improve the reliability and energy-efficiency of the asynchronous wake-up primitive, Blitz integrates a novel wake-up classification scheme for mitigating erroneous wake-ups, a phenomenon commonly associated with the use of wake-up receivers. We present a prototype implementation of Blitz using a wireless embedded platform designed according to the Bolt platform architecture, which incorporates the ultra-low power wake-up receiver integrated on the Zippy platform. We experimentally evaluate the performance of the proposed wake-up classification technique and the responsiveness of Blitz in an indoor testbed deployment, and evaluate the energy efficiency of Blitz in the context of a wireless acoustic emission sensing application. We show that Blitz achieves a latency of 108.9 ms across 4 hops, while dissipating only 16 μW during periods of inactivity.

5.1 Introduction

Motivation. Event-triggered wireless sensing systems are an important class of wireless sensor network, whereby a spatially distributed network of resource constrained source nodes detects *events* which are rapidly communicated to a remote host for analysis. Examples of these systems include surveillance systems [ZFB10], industrial monitoring [CEP16], and early warning systems for natural hazards [GBG+12].

In such systems, *events* arrive according to a *non-deterministic* arrival rate. Once an event is detected by an application-specific sensor, the source node extracts the sensor *data* associated with the event, possibly pre-processes the data, before acquiring appropriate network bandwidth and disseminating the event and its associated data to the remote host. Examples of events and their associated data include the detection of an intruder using an infrared sensor coupled with a picture of the intruder, the detection of a gas leak using a gas sensor coupled with the hydrocarbon response of the air sample, and the detection of a rock wall fracture using an acoustic sensor coupled with the frequency response of the acoustic emission. Once the host analyzes the event and its associated data, it instructs the network of source nodes on the appropriate action, for example, turning on additional sensor modalities, controlling an actuator, or to disregard the event and await the arrival of the next event.

In order for the remote host to react quickly to an event, the underlying communication architecture of the event-triggered wireless sensing system must support the following functional requirements:

- ***Responsiveness:*** Events and their associated data must disseminate through a multi-hop network with minimal latency. A dissemination latency on the order of hundreds of milliseconds make it possible to meet the requirements of a wide range of application domains [BKK15].

- *Energy Efficiency:* The energy consumption of all source nodes and the host must be minimized in order to maximize the operational lifetime. An average power dissipation of microwatts during periods of inactivity is necessary to support operation for multiple years using a low-capacity battery.

In this chapter, we present a novel communication architecture that exhibits these functional requirements *simultaneously*, thereby supporting efficient wireless communication for event-triggered wireless sensing systems.

Challenges. The dissemination of events to a remote host may be achieved using *periodic communication*, i.e., using pseudo-asynchronous protocols such as LPL [PHC04], LPP [MELT08] or variants thereof, or synchronous protocols such as the LWB [FZMT12], where nodes communicate according to a specified radio duty-cycle. However, this communication scheme exhibits a fundamental trade-off between latency and energy efficiency [MDN13]. When an event is detected, a node must wait until the beginning of its next communication round before dissemination commences, thereby increasing latency and adversely impacting responsiveness. Furthermore, since all nodes must communicate periodically to maintain network state, precious energy resources are expended irrespective of the detection of an event. One may improve energy efficiency by communicating less frequently, but only at the cost of increasing latency. This fundamental trade-off is a severe design constraint [DGA+05], that until recently, has received little attention in the literature [SSV12, HSR16].

Approach. We overcome this fundamental trade-off by facilitating *event-triggered communication*, where nodes in a multi-hop network only communicate when there is an event to disseminate, therefore conserving precious energy resources between event arrivals. We propose Blitz, a novel communication architecture that combines two orthogonal communication primitives using interference-based flooding. As illustrated in Figure 5.1, we consider a multi-hop network of resource-constrained *source* nodes S_i and a remote *host* H. During periods of inactivity, i.e., when no events are detected, the entire network resides in a *deep sleep listening* state where energy consumption is at its lowest, while continuously listening to the wireless channel using an ultra-low power wake-up receiver. When an event is detected, Blitz invokes two communication primitives, namely, *(i)* asynchronous wake-up, and *(ii)* synchronous dissemination. The asynchronous wake-up primitive

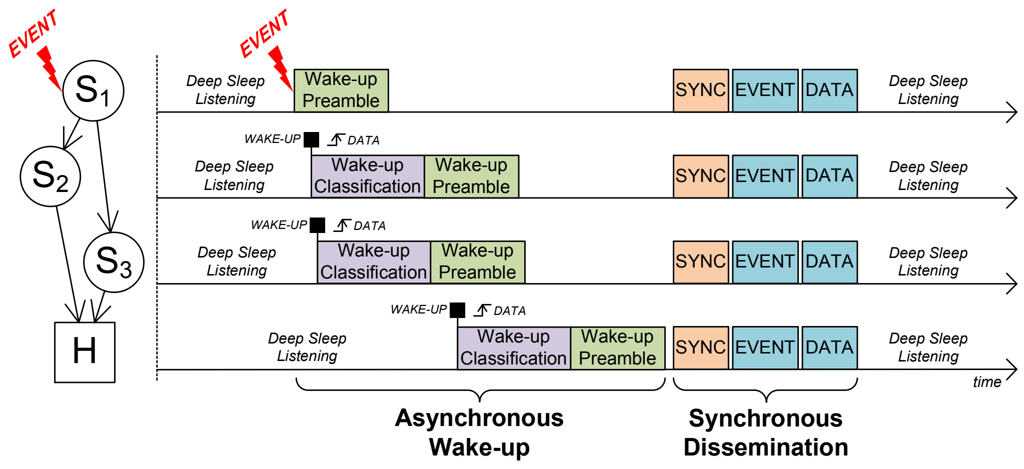


Figure 5.1: Example multi-hop network and the radio activity of BLITZ when source node S_1 disseminates an event and its associated data to the host H.

quickly awakes the network hop-by-hop from the energy conserving state, before the synchronous dissemination primitive synchronizes the network, arbitrates network bandwidth, and disseminates the event and its associated data to the host. The entire network then returns to the deep sleep listening state until the next event is detected.

The BLITZ communication architecture utilizes *interference-based flooding*, whereby the asynchronous wake-up and synchronous dissemination primitives employ network flooding, while allowing neighboring nodes to cause interference through simultaneous transmissions. These techniques may appear counterintuitive, since wake-up receivers are known to suffer from erroneous wake-ups that lead to energy inefficiencies [PMK+17], flooding potentially wastes energy through redundant transmissions [PCK16], and simultaneous transmissions may lead to packet corruption. But we will show that the proposed communication architecture facilitates event-triggered multi-hop communication without expending significant energy resources on erroneous wake-ups, without having to trade-off responsiveness and energy efficiency, and without suffering from excessive packet corruption.

Contributions. This chapter makes the following contributions:

- We present a novel communication architecture that supports low latency and energy-efficient event-triggered wireless communication using interference-based flooding.
- We propose a new scheme for mitigating the occurrence of erroneous wake-ups, which are commonly associated with ultra-low power wake-up receivers. We present an analytical model of the wake-up classifier, determine its power-optimal configuration, and

experimentally evaluate the performance of the proposed wake-up classifier in an indoor testbed deployment.

- We introduce an analytical model to quantify the limits of Blitz compared to a state-of-the-art protocol based on periodic communication.
- We present a prototype of Blitz and experimentally evaluate its performance with respect to latency in an indoor testbed, and energy efficiency in a laboratory setting.

5.2 Design of Blitz

Intuition. In order to motivate the design of Blitz, we first outline the intuition behind achieving efficient event-triggered wireless multi-hop communication:

> *Sleep as long as possible, quickly wake-up when there is something interesting to share, and promptly arbitrate information transfer.*

Blitz achieves the desired functional behavior through the combination of two orthogonal communication primitives. Specifically, the *asynchronous wake-up* primitive ensures that the entire network wakes up when there is something interesting to share, while minimizing the energy consumption during periods of inactivity. The *synchronous dissemination* primitive then promptly arbitrates the available network bandwidth such that information is transferred between source nodes and the host.

We next detail the design of each communication primitive in turn, before describing how the two primitives interact to achieve low latency and energy-efficient event-triggered multi-hop communication.

5.2.1 Asynchronous Wake-up

Requirements. In order for all source nodes in the network to sleep, i.e., conserving energy resources by operating in a low-power mode, for as long as possible, each source node must only wake-up if either *(i)* it has detected an event to disseminate to the host, or *(ii)* another source node in the network has detected an event to disseminate to the host.

Challenges. Due to the non-deterministic nature of event arrivals, source nodes do not know in advance when they will detect an event, and nor do they know when any other source node in the network will detect an event. A common approach in the literature is to employ a periodic communication scheme, however this leads to an undesirable design trade-off between responsiveness and energy efficiency.

Proposal. An alternative approach is to employ an on-demand wake-up scheme facilitated by wake-up receivers [GKSR12]. Wake-up receivers are ultra-low power low-complexity demodulation circuits capable of detecting a carrier signal. The power dissipation of wake-up receivers is typically on the order of microwatts, which is low enough to achieve an operational lifetime of several years using a low-capacity coin cell battery. An always-on wake-up receiver enables a node to wake-up its one-hop neighbors at any time by transmitting a wake-up preamble, i.e., a burst of a carrier signal. The wake-up receiver detects the wake-up preamble and produces a digital output that can be used by the node to awaken it from the deep sleep listening state.

In order to quickly wake-up a multi-hop network, BLITZ floods wake-up preambles. As soon as a node receives a wake-up preamble, it immediately transmits a wake-up preamble to wake-up its one-hop neighbors. After some time, all nodes in the multi-hop network will be awake and ready to proceed with event and data dissemination. A practical limitation of this approach is that the simultaneous transmissions from neighboring nodes may destructively interfere, resulting in either a delayed or a missed wake-up. Rather than attempting to avoid simultaneous transmissions of wake-up preambles, we instead encourage the interference by randomizing the structure of the wake-up preamble transmission such that the probability of complete destructive interference is reduced, while taking advantage of the superposition of signals from neighboring nodes to improve reliability. We employ the Carrier Frequency Randomization technique, as introduced and experimentally evaluated in Chapter 4, to randomize the structure of the wake-up preambles.

While ultra-low power wake-up receivers facilitate rapid asynchronous wake-up, their low-complexity receiver structures makes them susceptible to interference. Interference sources can cause a significant waste of energy, as a node will mistakenly transmit a wake-up preamble, turn on its high-powered transceiver, and proceed with the synchronous dissemination primitive. In order to mitigate this significant loss of energy, we incorporate a novel method for distinguishing between *erroneous wake-ups* and *correct wake-ups*, and therefore transmit the wake-up preamble and start the synchronous dissemination primitive only when a correct wake-up is identified.

5.2.2 Synchronous Dissemination

Requirements. Once all nodes in the network are awake, the available network resources must be arbitrated based on *(i)* which nodes have an event to disseminate, and *(ii)* how much bandwidth is required to disseminate each event and its associated data.

Challenges. The key challenge is that when the network awakes from the deep sleep listening state, there is no network state to take advantage of. Specifically, there is no local or global time synchronization, the network topology is unknown, and the bandwidth requirements between source nodes and the host are yet to be determined.

Proposal. In principle, one may use any multi-hop protocol from the literature [HXS+13] to acquire network state and deliver the event and data to the host. However, the time and energy required to acquire the necessary network state will impact overall responsiveness and energy efficiency. We therefore chose a *synchronous* and *topology-agnostic* protocol. This unique combination of properties takes advantage of a globally-synchronized schedule for rapid bandwidth arbitration without having to spend time and energy to discover the network topology.

As has been shown in Glossy [FZTS11], commodity low-power transceivers can be used to achieve topology-agnostic global time synchronization by exploiting constructive interference. Through careful time-triggered operation of the transceiver, the transmission of symbols can be aligned so that they constructively interfere, thus improving the reliability of packet reception. When constructive interference is combined with network flooding, which is referred to as a Glossy flood [FZTS11] in the literature, one can quickly synchronize a multi-hop network with a fine-grained resolution.

The Event-based Low-power Wireless Bus (eLWB), as presented in Chapter 3, is a synchronous protocol that uses time-slotted Glossy floods for the dissemination of events and their data. Since the eLWB combines topology-agnostic synchronization with adaptable bandwidth allocation, we chose to integrate it into the Blitz communication architecture.

5.2.3 Blitz Overview

The Blitz communication architecture facilitates the dissemination of an event, and its associated data, by enacting the asynchronous wake-up and synchronous dissemination primitives in sequence, as illustrated in Figure 5.2. In order to reduce complexity, we explain this interaction using a single-hop network between a source and a host node, and defer

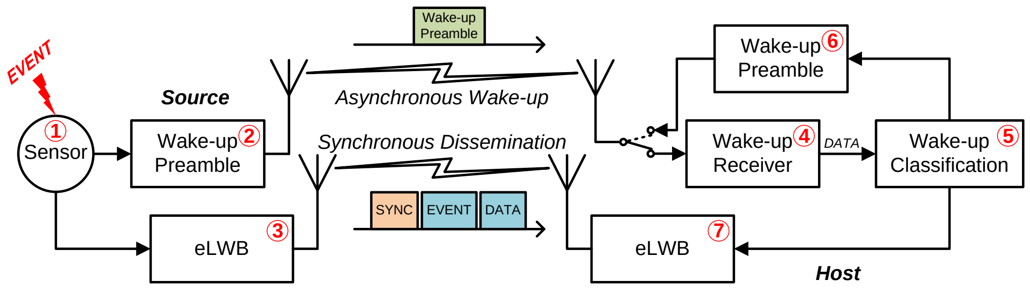


Figure 5.2: The interaction between asynchronous wake-up and synchronous dissemination primitives during a Blitz dissemination.

the description of intermediate source nodes to Section 5.4.

As depicted in Figure 5.2, when an event is detected at the source node using its attached sensor *(1)*, the node awakes from the deep sleep listening state and immediately transmits a wake-up preamble *(2)* so to asynchronously wake-up the host. The source node also starts the eLWB *(3)* in anticipation for the synchronous dissemination of the event and its associated data.

A short time later, the wake-up preamble is received by the host using its always-on wake-up receiver *(4)*. The digital output of the wake-up receiver, termed the *DATA* line (as defined in Section 4.2.2 of Chapter 4), follows the envelope of the wake-up preamble, i.e., the presence of a carrier signal is represented by a high *DATA* level, while the absence of a carrier signal is represented by a low *DATA* level. It follows that the rising-edge of the *DATA* line provides a trigger for the host to awake from the deep sleep listening state.

As will be detailed in the next section, Blitz incorporates a wake-up classification mechanism *(5)* that determines if the wake-up is legitimate, or if it is a consequence of a nearby interference source. If the wake-up is determined to be legitimate, the host immediately transmits a wake-up preamble *(6)* in order to flood the wake-up preamble through the rest of the network, and starts the eLWB *(7)*. As detailed in Chapter 3, the eLWB facilitates the dissemination of the event and its associated data using three rounds of interference-based flooding. Specifically, the eLWB facilitates synchronization to the host using a SYNC round, supports the dissemination of events and the request of network bandwidth using an EVENT round, and finally provides contention-free dissemination of the event's associated data using a DATA round. It is important to highlight that the structure of the eLWB rounds may be adapted to support a range of communication scenarios, including host-to-source unicast or broadcast. Once the synchronous dissemination is complete, the source and host nodes return to the deep sleep listening state.

5.3 Wake-up Classification

We next present the design of the wake-up classifier incorporated into the Blitz communication architecture.

Requirements. It is well known in the literature that wake-up receivers are susceptible to in-band and out-of-band interference sources due to the low-complexity receiver structures employed [DEO09]. When these interference sources are within range of the wake-up receiver antenna, the ultra-low power receiver circuit may detect a non-existent wake-up preamble, which we term an *erroneous wake-up*. The detection of an erroneous wake-up leads to a significant waste of energy, since the node will awake from the deep sleep listening state and and proceed with superfluous communication using its high-powered transceiver. This problem is further exacerbated using wake-up flooding, as employed by Blitz, since the detection of one erroneous wake-up will result in the entire network waking-up and wasting precious energy resources. We therefore require a scheme that can mitigate erroneous wake-ups, without adversely impacting the detection of *correct wake-ups*, i.e., wake-up preambles that were transmitted by a neighboring node for the purpose of a Blitz dissemination.

Challenges. While out-of-band interference may be rejected using passive filtering, e.g., [HMH11] or active filtering, e.g., [CLY+12], in-band interference is more difficult to mitigate against. The most common approach is to append an address to the wake-up preamble, which is referred to as an addressable or selective wake-up in the literature [DEO09]. By extending the wake-up receiver with a hardware-based, e.g., [ORW13, ABD15] or software-based, e.g., [MJS+16] correlator, the wake-up receiver only indicates the detection of the preamble if the decoded address matches the preconfigured address of the wake-up receiver.

Addressable wake-ups effectively mitigate erroneous wake-ups in single-hop networks, however, this approach is not robust in multi-hop networks. When several nearby nodes wish to wake-up their neighbor, e.g., during the asynchronous wake-up primitive incorporated into Blitz, the address of the wake-up preamble will likely be corrupted, therefore failing to wake-up the intended node. The reason for this behavior is a combination of two fundamental characteristics of OOK-based wake-up receivers. First, the time to detect a wake-up preamble is non-deterministic, as evaluated in Chapter 4. This means that during a flooding sequence, several nearby nodes will transmit a wake-up preamble with a non-negligible relative time offset. Secondly, using OOK modulation, a 0-bit will be decoded as 1-bit if a nearby node simultaneously transmits a 1-bit. This results in time-shifted overlapping transmissions of the preamble address that will likely corrupt address decoding, and therefore prevent the wake-up of the intended node.

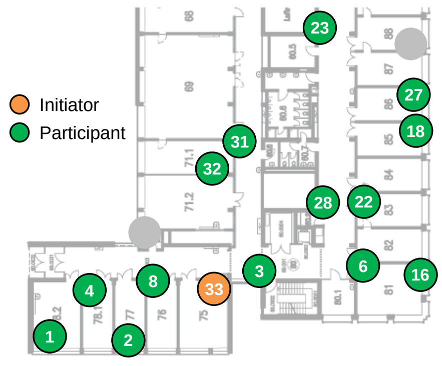
In principle, these limitations may be overcome by reducing the bit rate of the address decoding, and assigning the same address to all nodes in the network. Specifically, the bit rate would need to be low enough for the address decoder to handle all possible timing offsets between all one-hop neighbors in the network. Since the bit rate of the address decoder determines the per hop wake-up delay, implementing such a solution would severely impact the responsiveness of the asynchronous wake-up, and is therefore not considered a feasible solution.

Unfortunately, the use of FSK-based wake-up receivers, such as those presented in [LMvR14], do not overcome the aforementioned limitation of addressable wake-ups in multi-hop networks. The capture effect [LF76] may support the successful reception of multiple overlapping FSK transmissions, albeit only under specific conditions. As experimentally verified in [KW16], these conditions are determined by the relative time of arrival of each signal and the difference in relative signal powers. Additionally, novel schemes such as time-separated wake-ups [ODP+12], adaptive decision thresholding [ORW13], multi-band addressing [HKV+14, PSTT14], time/frequency addressing [dFZ11], and differential addressing [SURK17] do not circumvent the corruption of addressed wake-up preambles within multi-hop networks.

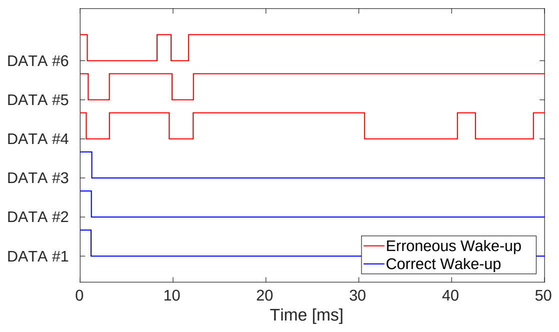
Proposal. We instead take an alternative approach, where we exploit the inherent structure of the wake-up receiver's digital output to determine if a detected wake-up preamble represents a *correct* or an *erroneous* wake-up. When a wake-up preamble is detected, we observe the output of the wake-up receiver, i.e., the *DATA* line, for a short period of time and extract appropriate time-domain features from the digital signal. We then use a pattern classifier to determine, within some probabilistic bounds, if the detected wake-up is correct, i.e., the high-powered transceiver is turned on and BLITZ dissemination proceeds, or if the wake-up is erroneous, i.e., the node returns to the deep sleep listening state and awaits the detection of the next wake-up preamble.

5.3.1 Ground Truth, Feature Selection & Classification

Data Collection. In order to investigate the time-domain characteristics of the wake-up receiver digital output, we performed a series of experiments on the FLOCKLAB [LFZ+13b] indoor testbed with fine-grained tracing capabilities [LMD+15] enabled. We used a network of 15 ZIPPY nodes deployed according to the map depicted in Figure 5.3(a), to collect ground



(a)



(b)

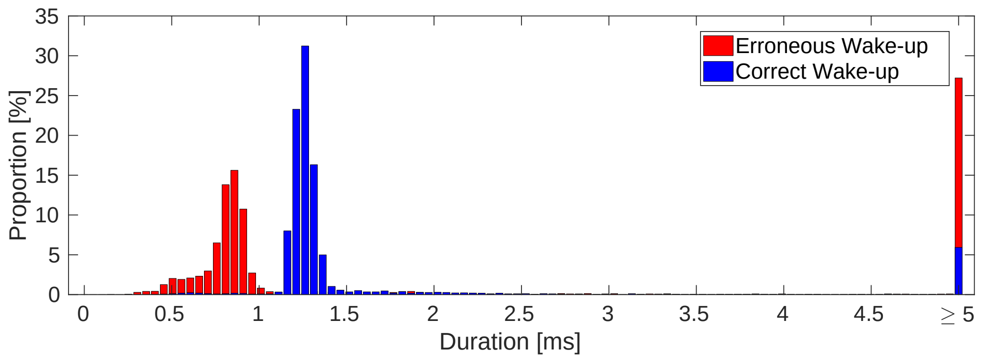
Figure 5.3: (a) Map of the FlockLab deployment used for the collection of erroneous and correct wake-ups, and (b) three example *DATA* signal traces from each wake-up class.

truth *DATA* signal traces for both erroneous and correct wake-up classes.

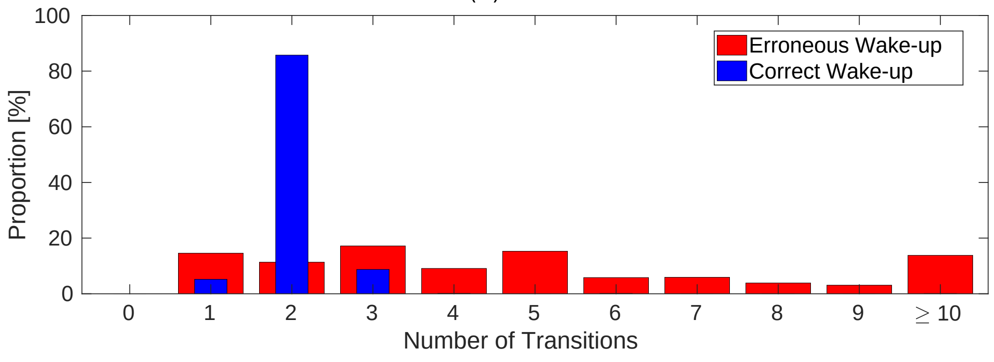
Due to practical constraints, the Zippy nodes were installed into FlockLab observers by stacking them on top of deployed Dual-Processor Platforms (DPP), which were evaluated in Chapter 3. Since the DPP antennas exhibit a non-zero S_{11} return loss at the input frequency band of the wake-up receiver, we observed a degraded link performance for the deployed Zippy nodes. Therefore, in order to improve the link budget of selected links, we integrated a HMC788A fixed-gain RF amplifier on nodes 33 and 3. This increased the maximum transmission power of these nodes from +10 dBm to +16.9 dBm, which is within the limits stipulated by the radio regulations of Switzerland, as defined in [Swi16].

We first collected erroneous wake-ups by turning on each wake-up receiver and awaiting for the *DATA* line to activate in response to an uncontrolled interference source within the environment. Once an erroneous wake-up was detected, the level of the *DATA* line was recorded for 50 ms before the wake-up receiver was reset. We then collected correct wake-ups by performing controlled periodic flooding of wake-up preambles, with node 33 configured as the initiator and all nodes configured with a fixed wake-up preamble duration of 1.4 ms. When a correct wake-up was detected, the level of the *DATA* line was recorded for 50 ms before the wake-up flooding continued. A total of 6693 *DATA* signal traces were collected for each wake-up class.

Feature Selection. In order to understand the differences between the two wake-up classes, we visualize a small set of signal traces for each wake-up class in Figure 5.3(b). In general, correct wake-ups typically exhibit only a single pulse with a duration similar to that of the wake-up preamble, i.e., specifically the duration of the wake-up preamble minus the detection time T_{wake} as defined in Chapter 4, while erroneous wake-



(a)



(b)

Figure 5.4: Histogram of (a) the duration of the first pulse, and (b) the number of transitions during the 50 ms *DATA* signal trace.

ups typically exhibit a shorter first pulse followed by several pulses of varying duration.

While these characteristics hold for most of the collected traces, there is a small percentage of signal traces that exhibit characteristics that closely match the alternate wake-up class. This is evident in the histogram of the first pulse duration and the number of transitions observed during the recorded *DATA* signal trace, as illustrated in Figure 5.4(a) and Figure 5.4(b), respectively. While there is a separation between the two classes at a first pulse duration of approximately 1.1 ms, there is at least 5% of correct and erroneous wake-ups that both have a pulse duration longer than 5 ms. Furthermore, there is at least 11% of correct and erroneous wake-ups that both exhibit a *DATA* signal trace containing two transitions. It became apparent through extensive testing and analysis that this undesirable feature overlap was caused by only a few nodes, specifically the nodes that were operating close to the limits of the wake-up receiver sensitivity.

Based on the time-varying nature of the *DATA* signal traces, we propose six time-domain features, as summarized in Table 5.1, to distinguish between correct and erroneous wake-up classes. In order to bound the classification time, we evaluate the continuous and discrete feature set within a fixed *classification window* of duration T_c.

Table 5.1: Discrete and continuous features for wake-up classification.

Feature	**Description**
X_1	Duration of the first pulse
X_2	Number of transitions
X_3	Arrival of second pulse
X_4	Duration of second pulse
X_5	Number of low to high transitions
X_6	Number of high to low transitions

Table 5.2: Performance metrics and their relation to the confusion matrix of the trained wake-up classifier.

	Predicted: *CORRECT*	Predicted: *ERRONEOUS*
Actual: *CORRECT*	$1 - p_m$	p_m
Actual: *ERRONEOUS*	p_f	$1 - p_f$

Classification. In order to evaluate the performance of a trained wake-up classifier, we define metrics that represent the desired behavior of the wake-up classifier in terms of misclassification. Intuitively, we seek to minimize the number of misclassified erroneous wake-ups and minimize the number of misclassified correct wake-ups. We therefore define *(i)* the probability of a false wake-up p_f, and *(ii)* the probability of a missed wake-up p_m, which represent the proportion of misclassified erroneous wake-ups and the proportion of misclassified correct wake-up given a fixed set of training data, respectively. These two probabilities are estimated from the confusion matrix of the trained classifier, as represented in Table 5.2.

We next train Decision Tree (DT) and Support Vector Machine (SVM) classifiers using the collected labeled *DATA* signal traces with 10-fold cross-validation and for a range of classifier windows $T_C \in [0, 4]$ ms. Figure 5.5 illustrates the performance of each classifier in terms of the probability of false and missed wake-ups. We can observe that the performance of the two classifiers is poor for $T_C < 1.3$ ms, but significantly improves as the observation window increases $T_C > 1.3$ ms. This behavior concurs with the structure of the histogram presented in Figure 5.4(a), as the highest proportion of correct wake-ups exhibit a first pulse duration of approximately 1.3 ms. It is also evident from Figure 5.5 that the performance of the two classifiers is comparable for $T_C > 1.3$ ms, since the probability of a missed wake-up converges at a similar rate, i.e., to approximately 9% at $T_C = 4$ ms, while the probability of a false wake-up remains below 9% for both classifiers. Since DT classifiers exhibit a lower

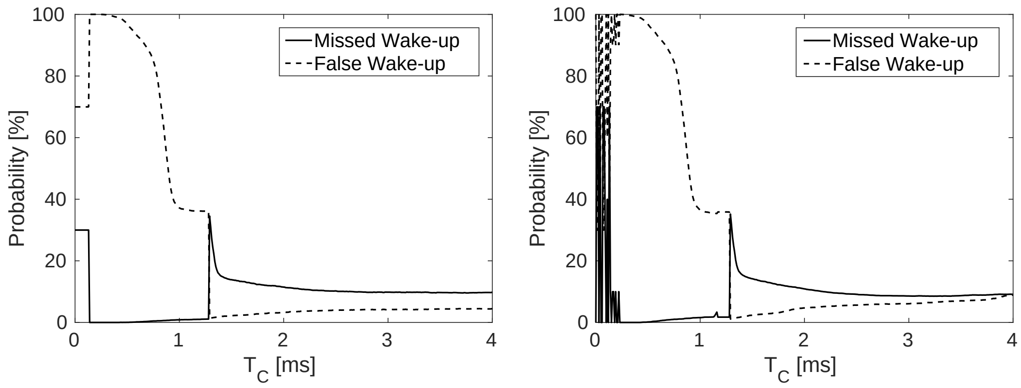


Figure 5.5: Performance of (left) DT and (right) SVM wake-up classifiers.

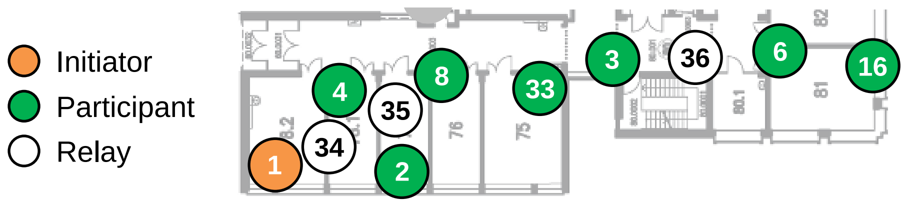
Table 5.3: Binary features used for the classification of wake-ups.

Binary Feature	Description
$Y_1(t_1, t_2)$	Duration of the first pulse within interval $[t_1, t_2]$
$Y_2(k)$	Number of transitions equal to k within T_C

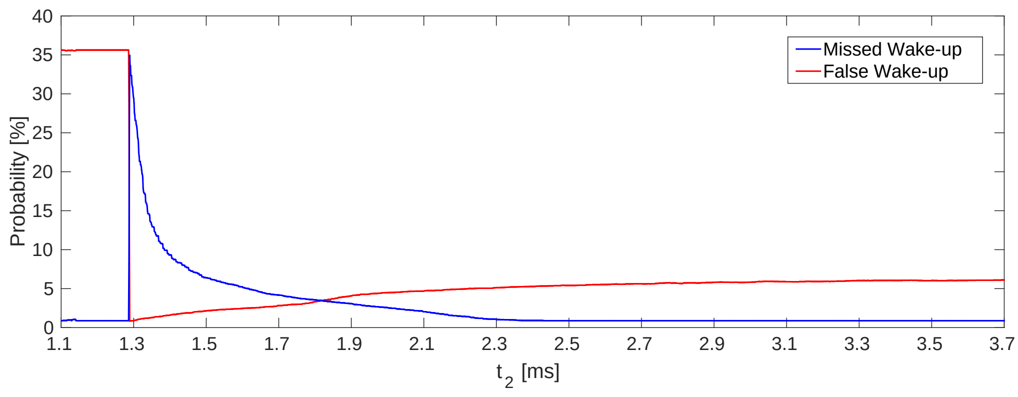
implementation complexity compared to SVM classifiers [DHS12], we choose to adopt the DT classifier for the remaining analysis.

Classification with Binary Features. A deeper inspection of the classification results shows that the duration of the first pulse X_1 and the number of transitions X_2 are the two most dominant features. Specifically, the majority of correct wake-ups are characterized by *(i)* a first pulse greater than 1.1 ms based on a 1.4 ms wake-up preamble duration, and *(ii)* contain only two transitions corresponding to the rising and falling edge of the detected wake-up preamble. We therefore reduce the dimensionality of the feature set to only two binary features, $Y_1(t_1, t_2)$, and $Y_2(k)$, as summarized in Table 5.3. We define feature $Y_1 = 1$ if the duration of the first pulse is within the interval $[t_1, t_2]$, and $Y_2(k) = 1$ if the number of transitions within the classifier window T_C is equal to k. If these conditions are not met, the respective feature is zero. Based on the analysis thus far, we parameterize the features with $t_1 = 1.1$ ms and $k = 2$, while t_2 is an unknown feature parameter that determines the performance of the trained wake-up classifier.

In order to evaluate the wake-up classifier using binary features, we deployed three relay nodes and omitted several leaf nodes that exhibited poor link performance, as illustrated in Figure 5.6(a). The relay nodes function identically to the Zippy nodes deployed on FlockLab, except that they are not controlled or monitored by the FlockLab test infrastructure. As detailed in Section 5.3.1, Zippy nodes 34, 33, and 3 were configured to transmit at a higher output power using an integrated power amplifier. We collected a new data set of controlled correct wake-ups, and together with the existing erroneous wake-up data set, trained a decision tree



(a)



(b)

Figure 5.6: (a) FlockLab deployment incorporating relay nodes, and (b) the decision tree classifier performance according to feature parameter t_2.

classifier using the aforementioned binary features for a range of values t_2, while setting the classification window $T_C = t_2$.

The performance of each decision tree classifier instance is illustrated in Figure 5.6(b). The results indicate that the improved link performance of the FlockLab deployment has made it easier to distinguish between correct and erroneous wake-ups, since the false and missed wake-up probabilities are significantly less than 9%. Secondly, it is evident that the value of the feature parameter t_2 determines if the wake-up classifier favors a lower missed wake-up probability, a lower false wake-up probability, or a equilibrium of the two probabilities. This raises the question of how one should best parameterize the wake-up classifier, as it is not obvious how to trade-off the impacts between false and missed wake-ups, since an increased false wake-up probability p_f wastes precious energy resources in superfluous Blitz dissemination, while an increased missed wake-up probability p_m adversely impacts the end-to-end reliability of the Blitz dissemination. We therefore introduce an analytical model that investigates this complex trade-off in terms of energy-efficiency and reliability of Blitz dissemination.

5.3.2 Wake-up Classifier Model

Model Description. We consider a network of source nodes S and one host node H arranged in a h-hop line topology, as depicted in Figure 5.7.

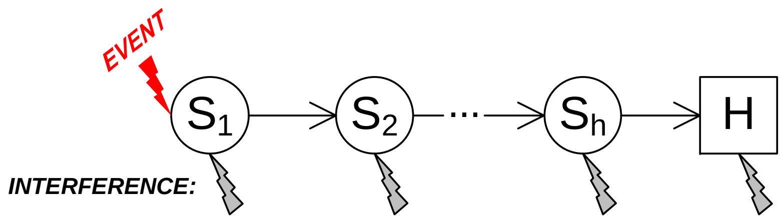


Figure 5.7: Network assumed for the wake-up classifier model. Node S_1 detects events, while all nodes are in the presence of an independent interference source.

We assume source node S_1 detects events with an average rate λ_c, and disseminates each event and its associated data to the host using BLITZ. All nodes in the network are subject to interference i.e., either in-band or out-of-band interference, with an average rate λ_e. It is assumed that the detection of interference contains no useful information for the host.

We next determine the energy consumption of BLITZ when it incorporates a wake-up classifier to distinguish between correct and erroneous wake-ups.

Energy Consumption of Correct Wake-ups. In order for source node S_1 to successfully disseminate an event and its associated data to the host, all nodes in the line topology must awake, i.e., the classifier in each node must predict the detected wake-up preamble to be a correct wake-up. However, as listed in Table 5.2, a node will classify a correct wake-up with a probability of $1-p_m$. Therefore, assuming well-connected links between nodes and that the successful detection of wake-ups are uncorrelated, the probability that S_1 successfully awakes the h-hop network is given by the following:

$$Pr\,(\text{successful wake-up flood}) = (1 - p_m)^h \tag{5.1}$$

To give a concrete example, in order to successfully wake-up a 4-hop network 99.0% of the time, the missed wake-up probability must be less than 0.25%. Based on testbed experiments presented earlier, this is extremely difficult to achieve in practice since there is a non-negligible probability that a *DATA* signal trace from the erroneous wake-up class exhibits identical features of a trace from the correct wake-up class.

We circumvent this limitation by introducing redundancy into the BLITZ asynchronous wake-up primitive. Once S_1 initiates a wake-up flood in response to the detection of an event, it attempts to synchronize with the host using its high-powered transceiver. If after some fixed time, synchronization to the host is not achieved, S_1 will retransmit the wake-up preamble. We assume that S_1 may transmit a maximum of N wake-up preambles per event. We now derive in (5.2) the probability $P_{success}(N)$ that S_1 successfully awakes a h-hop network using at most $N \geq 1$ transmissions.

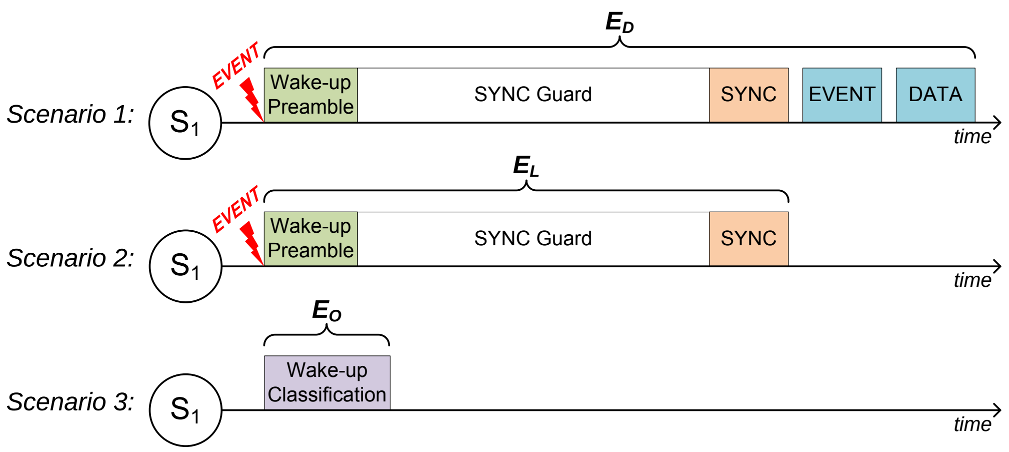


Figure 5.8: The energy consumption E_D for a successful Blitz dissemination, the energy loss E_L associated with a failed Blitz dissemination, and the energy overhead E_O for classifying a wake-up.

$$\begin{aligned} P_{success}(N) = & Pr(1^{\text{st}}\text{ success})+ \\ & Pr(1^{\text{st}}\text{ fail, }2^{\text{nd}}\text{ success})+ \\ & Pr(1^{\text{st}}\text{ fail, }2^{\text{nd}}\text{ fail, }3^{\text{rd}}\text{ success}) + ... \\ & Pr(\text{(N-1) failures, N}^{\text{th}}\text{ success}) \\ = & (1-p_m)^h \sum_{i=0}^{N-1} \left(1-(1-p_m)^h\right)^i \end{aligned} \tag{5.2}$$

The introduced redundancy makes it possible to achieve a higher level of reliability. For example, achieving a wake-up flooding reliability of 99.5% across a 4-hop network with at most $N = 3$ transmissions requires a missed wake-up probability less than 4.5%, i.e., one order of magnitude larger compared to without redundancy. As illustrated earlier in Figure 5.6(b), this is indeed feasible in practice using a decision tree classifier with two binary features.

However, the increase in reliability is accompanied by an increase in the average energy consumption of node S_1. We use Figure 5.8 to illustrate the three scenarios in which energy is consumed by S_1 during a Blitz dissemination, while deferring a detailed description of the Blitz protocol sequence to Section 5.4. We denote the energy consumption of S_1 successfully disseminating an event by E_D, the energy loss associated with a failed Blitz dissemination by E_L, and the energy overhead of classifying a wake-up by E_O. Since node S_1 is the node furthest away from the host, it then follows that the worst-case average energy consumption of correct

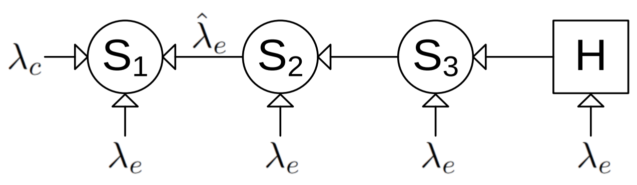


Figure 5.9: Average arrival rate of events λ_c, local interference λ_e, and interference originating from all other nodes in the network $\hat{\lambda}_e$.

wake-ups for $N = 3$ is given by the following:

$$\begin{aligned} E_c =& Pr(1^{\text{st}} \text{ success}) \times E_D + \\ & Pr(1^{\text{st}} \text{ fail}) \times Pr(2^{\text{nd}} \text{ success}) \times (E_L + E_D) + \\ & Pr(1^{\text{st}} \text{ fail}) \times Pr(2^{\text{nd}} \text{ fail}) \times Pr(3^{\text{rd}} \text{ success}) \times (2 \cdot E_L + E_D) + \\ & Pr(1^{\text{st}} \text{ fail}) \times Pr(2^{\text{nd}} \text{ fail}) \times Pr(3^{\text{rd}} \text{ fail}) \times (3 \cdot E_L) \end{aligned} \tag{5.3}$$

In the general case for $N \geq 1$, the worst-case average energy E_c for handling correct wake-ups, misclassified or not, is given by the expression in (5.4).

$$E_c = \left(1 - (1 - p_m)^h\right)^N NE_L + (1 - p_m)^h \sum_{i=0}^{N-1} \left(1 - (1 - p_m)^h\right)^i (iE_L + E_D) \tag{5.4}$$

Energy Consumption of Erroneous Wake-ups. There are two ways in which node S_1 consumes energy as a consequence of erroneous wake-ups. As depicted in Figure 5.9, the energy consumption is caused by *(i)* local interference at S_1 having an average arrival rate of λ_e, and *(ii)* interference originating from all other nodes in the network having an average arrival rate of $\hat{\lambda}_e$.

In the case of local interference, the energy consumption of S_1 is determined by the classification performance of erroneous wake-ups. Specifically, S_1 consumes energy E_L by participating in a superfluous Blitz dissemination due to a misclassified erroneous wake-up, as depicted in Figure 5.8(Scenario 2). However, if the erroneous wake-up is predicted by the wake-up classifier, S_1 only consumes the energy to execute the wake-up classifier, as denoted by E_O in Figure 5.8(Scenario 3). Therefore, the average energy E_e consumed by S_1 for handling erroneous wake-ups associated with local interference is given by expression (5.5).

$$E_e = p_f E_L + (1 - p_f) E_O \tag{5.5}$$

In the case of interference originating at any other node in the network, the energy consumed by S_1 is determined by the classification performance of correct wake-ups. This is due to the fact that once a node misclassifies an erroneous wake-up, i.e., with probability p_f, it will transmit a wake-up preamble exhibiting the features of a correct wake-up. The neighboring node will then participate in the wake-up flood if the correct wake-up is predicted by the classifier, i.e., with probability $1 - p_m$. This behavior repeats through the network, resulting in an average arrival rate of erroneous wake-ups $\hat{\lambda}_e$ at node S_1, as given by (5.6).

$$\hat{\lambda}_e = \sum_{i=1}^{h} (1 - p_m)^i p_f \lambda_e \tag{5.6}$$

Therefore, the average energy E_e consumed by S_1 for handling erroneous wake-ups generated by all other nodes in the network, misclassified or not, is given by (5.7).

$$\hat{E}_e = (1 - p_m)E_L + p_m E_O \tag{5.7}$$

Average Power Dissipation. We now multiply the average arrival rates and the average worst-case energy consumptions for handling correct and erroneous wake-ups, to yield the average power dissipation P_{avg} of Blitz dissemination utilizing a wake-up classifier, as given by (5.8).

$$P_{avg} = \lambda_c E_c + \lambda_e E_e + \hat{\lambda}_e \hat{E}_e \tag{5.8}$$

5.3.3 Power-optimal Wake-up Classification

We now use the analytical model presented in Section 5.3.2 to find a wake-up classifier configuration that minimizes the power dissipation of Blitz dissemination. Specifically, we seek to find the classifier configuration that minimizes the average power dissipation of Blitz dissemination P_{avg}, subject to a minimum wake-up flooding reliability R. Since parameters $t_1 = 1.1$ ms and $k = 2$ are already determined, we are left to parameterize the upper interval of the first pulse duration t_2, and the duration of the observation window $T_C \geq t_2$. As we assume that the execution time of the decision tree is negligible compared to the duration of the observation window, we therefore equate the observation window duration with the unknown feature parameter, i.e., $T_C = t_2$. The optimal classifier configuration is then determined by finding the observation window T_C that minimizes P_{avg}, while supporting $P_{success} \geq R$.

Using the prototype Blitz implementation, which is detailed later in Section 5.5, we parameterize the wake-up classifier model according to the values summarized in Table 5.4. Additionally, we used the ground truth erroneous data set to evaluate the worst-case average arrival rate of erroneous wake-ups, which we closely approximate by $\lambda_e = \frac{1}{60}\,\mathrm{s}^{-1}$.

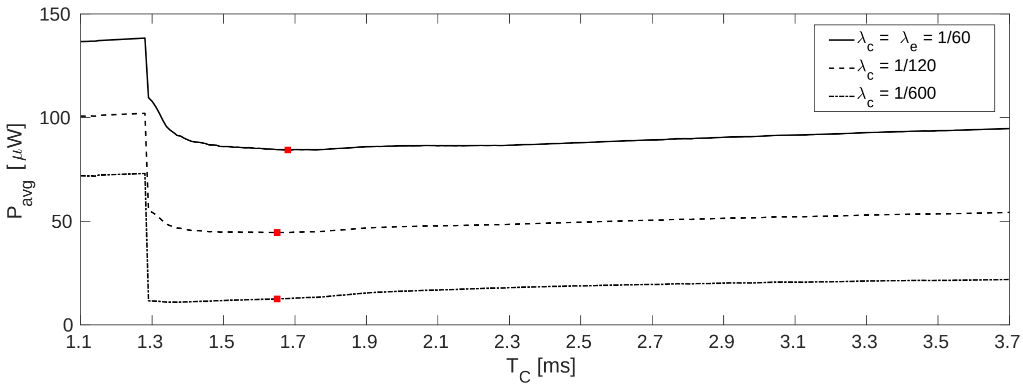


Figure 5.10: Power-optimal wake-up classifier configuration supporting $P_{success} \geq 99.5\%$, for an average event arrival rate of one event per minute, two minutes and ten minutes.

Table 5.4: Parameterization of the wake-up classifier model.

Parameter	Description	Value
h	Number of hops in the line topology	4
N	Maximum number of transmissions by S_1 per event	3
λ_c	Average arrival rate of correct wake-ups at S_1	$[\frac{1}{60}, \frac{1}{120}, \frac{1}{600}]s^{-1}$
λ_e	Average arrival rate of erroneous wake-ups	$\frac{1}{60}$ s $^{-1}$
E_D	Energy consumption of a successful dissemination	4.3 mJ
E_L	Energy consumption of an unsuccessful dissemination	2.3 mJ
E_O	Energy consumption for classifying a wake-up	3.5 μJ

We note that this is an extremely high rate of erroneous wake-ups for the indoor FlockLab deployment, but we use this value in order to determine an optimal wake-up classifier configuration under worst-case operating conditions.

Results. Figure 5.10 illustrates the average power dissipation of Blitz dissemination P_{avg} for three average event arrival rates λ_c. The power-optimal wake-up classifier configurations each supporting $R = 99.5\%$ are highlighted by a square marker. We note that the optimal value of T_C is similar for all three arrival rates, however, the average power dissipation increases as the average event arrival rate increases, which is the expected behavior.

The results also highlight that the power-optimal configuration point of the wake-up classifier is less than the equilibrium of p_m and p_f, as illustrated in Figure 5.6(b). This means that for the presented parameter set, the average power dissipation of Blitz dissemination is minimized when the wake-up classifier exhibits a false wake-up probability lower than the missed wake-up probability, i.e., $p_f < p_m$. In other words, the

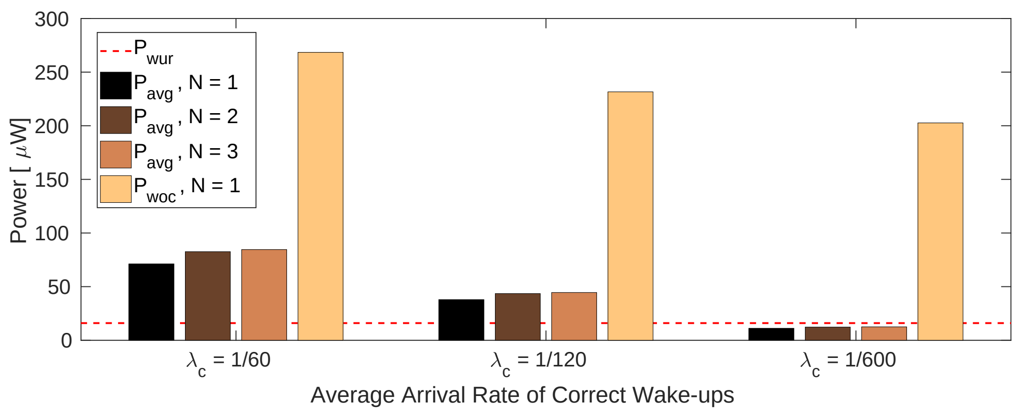


Figure 5.11: Comparison of P_{avg} to the average power dissipation of Blitz without a wake-up classifier P_{woc}, and the wake-up receiver P_{wur}. We note that P_{avg} does not include the power dissipation of the wake-up receiver P_{wur}.

analysis shows that Blitz dissemination is more energy-efficient when the energy consumption associated with erroneous wake-ups is reduced, despite the possibility of additional energy consumption through the retransmission of wake-up floods.

In order to put into perspective the magnitude of the Blitz average power dissipation, we compare P_{avg} for the power-optimal classifier configurations over a range of retransmissions N to the average power dissipation of *(i)* Blitz without a wake-up classifier P_{woc}, and *(ii)* the wake-up receiver P_{wur}, i.e., remaining in the deep sleep listening state. In the case where Blitz does not utilize a wake-up classifier, node S_1 will initiate a Blitz dissemination for all correct wake-ups, and will participate in all erroneous wake-ups generated by all other nodes in the network, resulting in an average power dissipation given by (5.9). As evaluated in Section 5.6, the developed prototype dissipates $P_{wur} = 16\,\mu$W during the deep sleep listening state.

$$P_{woc} = \lambda_c E_D + (h+1)\lambda_e E_L \tag{5.9}$$

As illustrated in Figure 5.11, P_{avg} increases only marginally as the number of retransmissions N is increased, and remains between 3.7× and 18.1× lower than the average power dissipation of Blitz without utilizing a wake-up classifier. Interestingly, depending on the specific value of λ_c, e.g., for $\lambda_c = \frac{1}{600}\,\text{s}^{-1}$, the average power dissipation of Blitz dissemination is actually lower than that of the wake-up receiver. While this may seem counterintuitive, this is a consequence of the relatively low average event arrival rate compared to the average arrival rate of erroneous wake-ups. In summary, Figure 5.11 emphasizes the significant energy savings achievable by incorporating the proposed wake-up classifier into the Blitz communication architecture.

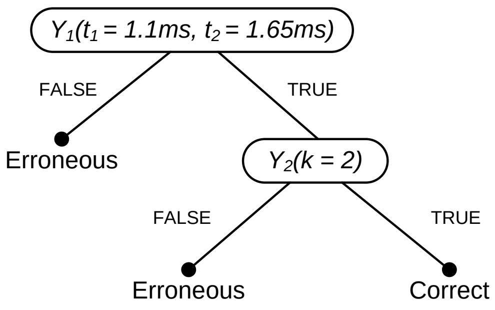


Figure 5.12: Decision tree corresponding to the power-optimal wake-up classifier which is experimentally evaluated in FlockLab.

5.3.4 Wake-up Classifier Evaluation

We next experimentally evaluate the wake-up classifier using the FlockLab deployment of Zippy nodes, as illustrated in Figure 5.6(a). We choose to evaluate the power-optimal wake-up classifier configuration for $\lambda_c = \frac{1}{120}\,\mathrm{s}^{-1}$, which corresponds to $T_C = 1.65\,\mathrm{ms}$. The decision tree of this particular configuration is depicted in Figure 5.12, and exhibits a false wake-up probability of 2.58%, and a missed wake-up probability of 4.56%. We implemented the wake-up classifier on all Zippy nodes and configured node 1 to periodically initiate 100 wake-up floods over a period of 50 minutes.

Setup. As we have no ground truth on the generation of erroneous wake-ups throughout the indoor deployment, we are unable to extract reliable statistics on the false wake-up probability. However, using the GPIO tracing feature of FlockLab, we can estimate the missed wake-up probability of each node. We achieved this by recording *(i)* the time of each initiated wake-up flood, *(ii)* the time associated with the rising edge of the *DATA* line, i.e., the beginning of classification, and *(iii)* the prediction made by the wake-up classifier. By comparing the time difference between the beginning of classification at each node with the time of the initiated wake-up flood, and applying a fixed threshold dependent on the maximum number of hops, we determine if a node classifies or misclassifies a correct wake-up. All correct wake-ups occurring outside this aforementioned time window are considered misclassified erroneous wake-ups. We note that some source nodes were not able to classify all initiated wake-up floods, i.e., due to a poor RF link or a neighbor misclassifying a correct wake-up. We therefore computed the number of classification attempts by each node, and used this to estimate the missed wake-up probability.

Results. As illustrated in Figure 5.13(a), all source nodes, apart from nodes 6 and 33, exhibited a missed wake-up probability less than that stipulated by the analytical results. The unusually high p_m values for

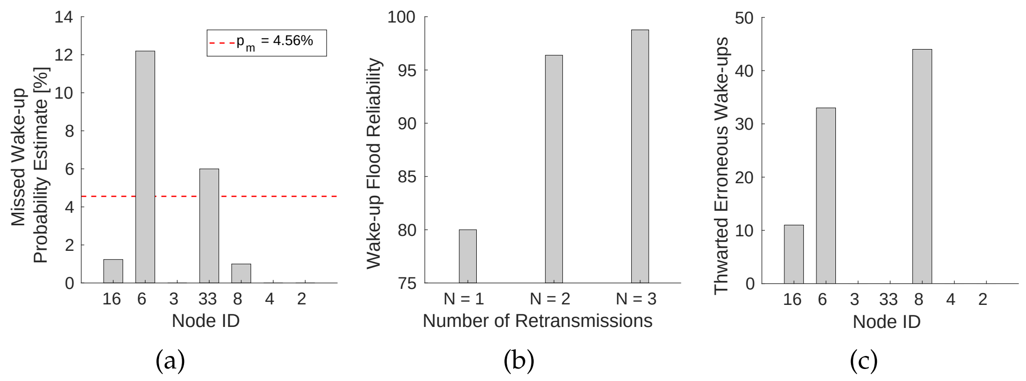


Figure 5.13: Evaluation of a power-optimal wake-up classifier configuration in FlockLab with respect to the (a) missed wake-up probability estimate, (b) wake-up flood reliability, and (c) number of thwarted erroneous wake-ups.

nodes 6 and 33 are due to a large number of misclassified correct wake-ups. An inspection of the corresponding *DATA* signal traces revealed that either the duration of the first pulse was too short, or a third transition occurred within the observation window. Due to the lack of observability of the relay nodes, it is difficult to isolate the root cause of this behavior. Nevertheless, for the majority of the nodes, the wake-up classifier performed within the expected bounds.

Using the FlockLab test data, we evaluated the wake-up flood reliability $P_{success}(N)$ if different levels of redundancy would be utilized. As shown in Figure 5.13(b), the reliability significantly increases from 80% with $N = 1$, to 98.7% with $N = 3$. Despite the poor performance with $N = 1$, which is mainly attributed to the misclassified wake-up of nodes 6 and 33, the host may be reached with high probability using an appropriate number of retransmissions.

Finally, we enumerated the number of times each node would have erroneously awoken, had the wake-up classifier not been employed. As shown in Figure 5.13(c), a total of 88 erroneous wake-ups were thwarted during the 50 minute evaluation on FlockLab. Assuming a Blitz parameterization according to Table 5.4, the classification of these erroneous wake-ups would consume $88 \times E_O = 0.4\,\text{mJ}$, compared to $88 \times E_L = 220\,\text{mJ}$ had a wake-up classifier not been employed. To put this into perspective, the resulting energy saving of 219.6 mJ would be sufficient to keep the wake-up receiver active for more than 3.5 hours, thus emphasizing the energy gains of incorporating the proposed wake-up classifier into the Blitz communication architecture.

5.4 Model-driven Comparative Analysis

We next present an analytical model to quantify the limits of Blitz. We compare the performance of Blitz to the eLWB presented in Chapter 3, as it is a representative of a class of protocols that are competitive with respect to latency, energy-efficiency and reliability [LFST17].

Model Description. As depicted in Figure 5.7, we consider a network of source nodes S and a host H arranged in a h-hop line topology. We assume node S_1, located h hops away from the host, detects an event and wishes to disseminate it together with its associated data to the host. The event arrival process at S_1 is assumed to be sporadic, and is characterized by the smallest time between sequential event arrivals T_{event}.

We assume that all nodes in the network adopt a multi-radio platform architecture consisting of a microcontroller interfaced to a transceiver and a wake-up receiver. The platform dissipates power P_s when the microcontroller and transceiver are in sleep mode, P_p when the microcontroller and transceiver are active, and P_{wur} when the wake-up receiver is active with the microcontroller and transceiver in sleep mode. It is assumed that the power dissipation for the transmission and reception of packets, and the transmission of a wake-up preamble, are equal.

Performance Metrics. In order to quantify the advantages and limitations of Blitz compared to the eLWB, we consider the following performance metrics:

- Best-case and worst-case latency, L_{min} and L_{max}, is the minimum and maximum time between the detection of a sporadic event at S_1 and the reception of the event and its associated data at the host H, respectively.
- Energy efficiency is represented by the average energy consumption per event E_{avg}, as evaluated at source node S_1.

We next detail the behavior of the eLWB and the Blitz protocol. In order to simplify the description, we consider a line topology with $h = 2$ hops, however the concepts presented are equally applicable to dense multi-hop topologies.

eLWB. The interaction between source and host using the eLWB is illustrated in Figure 5.14. The eLWB supports three types of rounds, namely SYNC, EVENT, and DATA rounds. The eLWB maintains synchronization using periodic SYNC rounds with duration T_S, according to a constant period $T_{round} \leq T_{event}$. In this model, the SYNC round period is constrained to ensure there is at least one opportunity to disseminate between the smallest event interarrival time T_{event}. When an event is

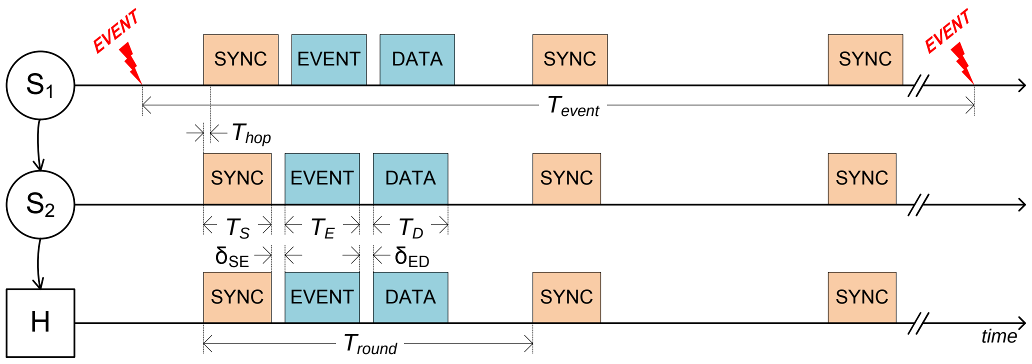


Figure 5.14: Radio activity using eLWB when node S_1 disseminates an event and its associated data to the host.

detected, the source node requests the host to schedule an EVENT round with duration T_E. During the EVENT round, the source disseminates the event to the host and requests bandwidth for the associated data. The host then schedules a DATA round with duration T_D for data dissemination. A detailed description of the round structure and the time separation between rounds, δ_{SE} and δ_{ED}, is presented in Chapter 3.

The eLWB rounds are implemented using time-slotted Glossy floods, whereby the transceiver sequentially receives a packet and immediately retransmits the packet a fixed number of times. Since the number of retransmissions is constant for all nodes in the network, the transceiver on time for each round is equal for all nodes in the network. However, due to the multi-hop propagation of Glossy floods, nodes residing $h > 1$ hops away from the host must listen for a duration of $(h-1)T_{hop}$ until the round commences [Zim15].

The best-case latency occurs when an event arrives just as a SYNC round begins, while the worst-case latency will be T_{round} longer, as represented in (5.10) and (5.11), respectively. Since the SYNC round period is independent of event arrivals, there will be on average $\eta = \frac{T_{event}}{T_{round}}$ rounds per event, leading to an average energy consumption per event according to (5.12).

$$L_{min}^{\text{eLWB}} = (h-1)T_{hop} + T_S + \delta_{SE} + T_E + \delta_{ED} + T_D \tag{5.10}$$

$$L_{max}^{\text{eLWB}} = L_{min}^{\text{eLWB}} + T_{round} \tag{5.11}$$

$$E_{avg}^{\text{eLWB}} = \Big(\eta(h-1)T_{hop} + \eta T_S + T_E + T_D\Big) P_p + \Big(T_{event} - (\eta(h-1)T_{hop} + \eta T_S + T_E + T_D)\Big) P_s \tag{5.12}$$

Blitz. Figure 5.15 illustrates the interaction between source and host using the proposed Blitz communication architecture. Once an event

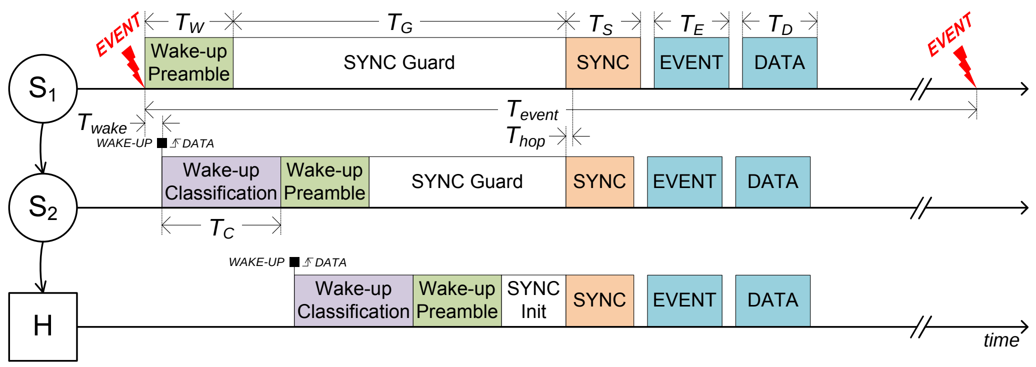


Figure 5.15: Radio activity using Blitz when node S_1 disseminates an event and its associated data to the host.

Table 5.5: Parameterization of the eLWB and Blitz analytical models.

Parameter	Value
T_S	22 ms
T_E	18 ms
T_D	22 ms
T_W	1.4 ms
T_{hop}	1.3 ms
T_{wake}	0.53 ms

Parameter	Value
T_{init}	15.24 ms
T_C	1.65 ms
δ_{SE}	6 ms
δ_{ED}	12 ms
P_s	8.5 μW
P_{wur}	16 μW
P_p	50 mW

is detected, the asynchronous wake-up primitive awakes the multi-hop network from the deep sleep listening state by flooding a wake-up preamble of duration T_W. Once the preamble transmission is complete, node S_1 must wait a guard time of T_G until the synchronous dissemination primitive commences. The guard time, as expressed in (5.13), is the sum of *(i)* the time to initialize the transceiver T_{init} and *(ii)* the time to execute the wake-up classifier T_C and to receive a wake-up preamble T_{wake}, accumulated over $h \geq 1$ hops.

$$T_G = T_{init} + h(T_C + T_{wake}) \tag{5.13}$$

Since the asynchronous wake-up primitive starts immediately after an event is detected, the best-case and worst-case latency for Blitz are identical. As expressed in (5.14), the latency is equal to the best-case latency of the eLWB plus the overhead for the asynchronous wake-up primitive. The average energy per event for Blitz is given by (5.15).

$$L_{min}^{\text{Blitz}} = L_{max}^{\text{Blitz}} = T_W + T_G + L_{min}^{eLWB} \tag{5.14}$$

$$\begin{aligned} E_{avg}^{\text{Blitz}} = &\Big(T_W + T_G + (h-1)T_{hop} + T_S + T_E + T_D\Big) P_p + \\ &\Big(T_{event} - (T_W + T_G + (h-1)T_{hop} + T_S + T_E + T_D)\Big) P_{wur} \end{aligned} \tag{5.15}$$

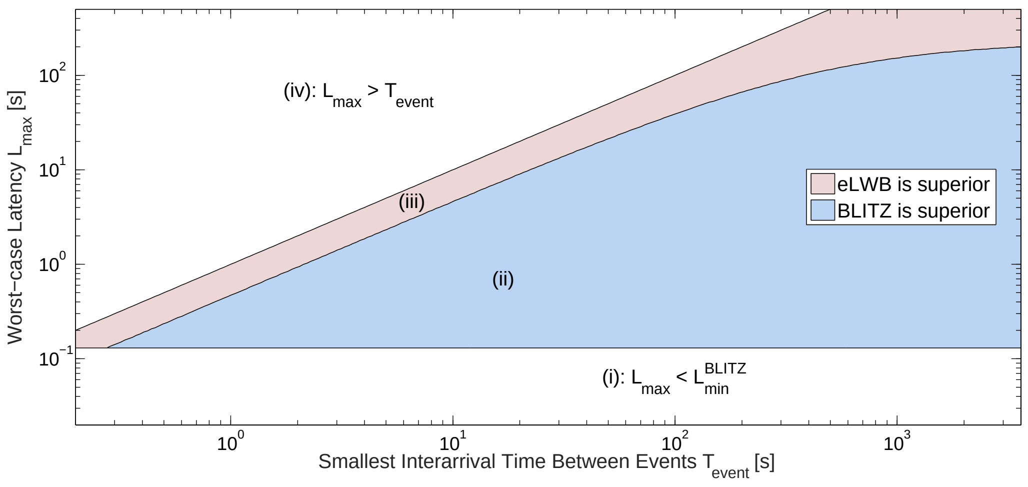


Figure 5.16: The application-specific conditions where Blitz consumes a lower average energy per event compared to using the eLWB, and where the eLWB consumes a lower average energy per event compared to using Blitz.

Results. Using measurements from a prototype implementation, we parameterize the analytical model according to the values listed in Table 5.5 for $h = 10$ hops. The analysis indicates that the latency for eLWB varies between 91.7 ms and 91.7 ms + T_{round}, while the latency for Blitz is constant at 130.1 ms. While the eLWB may achieve lower latency than Blitz in some specific cases, the eLWB worst-case latency will increase as T_{round} is increased so to reduce energy consumption. Blitz instead achieves a constant low latency of 130.1 ms, while only dissipating 16 μW during periods of inactivity.

We next investigate if the overhead associated with the Blitz asynchronous wake-up primitive improves energy efficiency. Given an application-specific worst-case latency, L_{max}, we evaluate the maximum eLWB round period according to equation (5.11), and then determine the T_{event} where the average energy per event for eLWB and Blitz are equal. Figure 5.16 illustrates the resulting non-linear protocol partitioning consisting of four unique regions. The first region represents where L_{max} is less than Blitz can support, the second region where Blitz is more energy-efficient, the third region where eLWB is more energy-efficient, and the fourth region where $L_{max} > T_{event}$, and is therefore infeasible under the model assumptions. The results highlight the superiority of Blitz over the eLWB in terms of latency and energy efficiency for a wide range of event-triggered application scenarios. The eLWB is preferred only when the constraint on worst-case latency is relaxed such that the periodic eLWB SYNC rounds consume less energy than keeping the wake-up receiver always on.

The superiority of Blitz compared to the eLWB, as shown in Figure 5.16, is an important result since it refines a common understanding

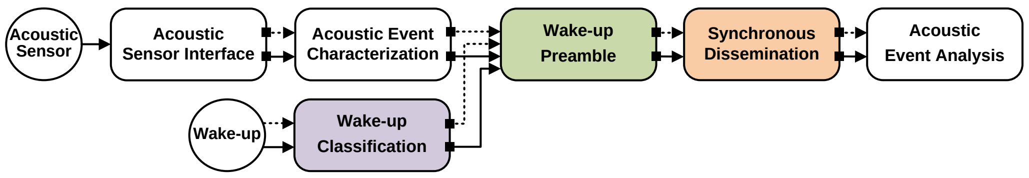


Figure 5.17: Logical event-triggered components of a wireless acoustic monitoring system incorporating the Blitz communication architecture.

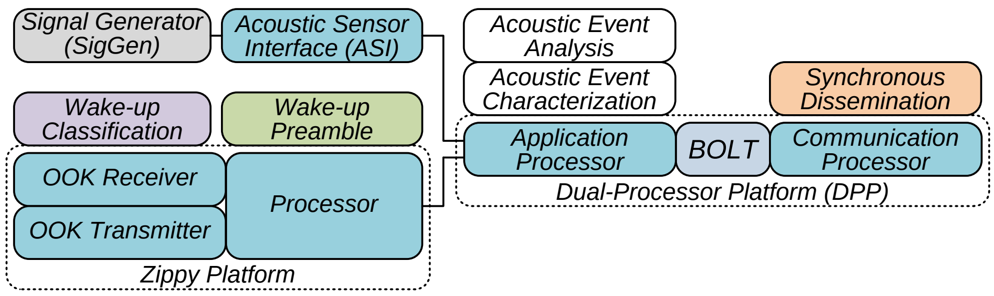
in the literature that wake-up receiver-based protocols are only more energy-efficient than duty-cycled protocols when the event arrival rate is low [ODC+16, AGM+17]. If event arrivals are deterministic, and assuming the model description depicted in Figure 5.7, then the duty-cycle of a synchronous or pseudo-asynchronous protocol could, at least in principle, always be tuned such that all nodes only communicate when events arrive. This would save significant energy and yield a latency dependent only on the implementation of the specific communication primitive employed. However, if event arrivals are non-deterministic, then the combination of asynchronous and synchronous communication primitives as used in Blitz yields a lower worst-case latency and lower energy per event compared to a state-of-the-art synchronous protocol for a wide range of event-triggered application scenarios.

5.5 Prototype Implementation

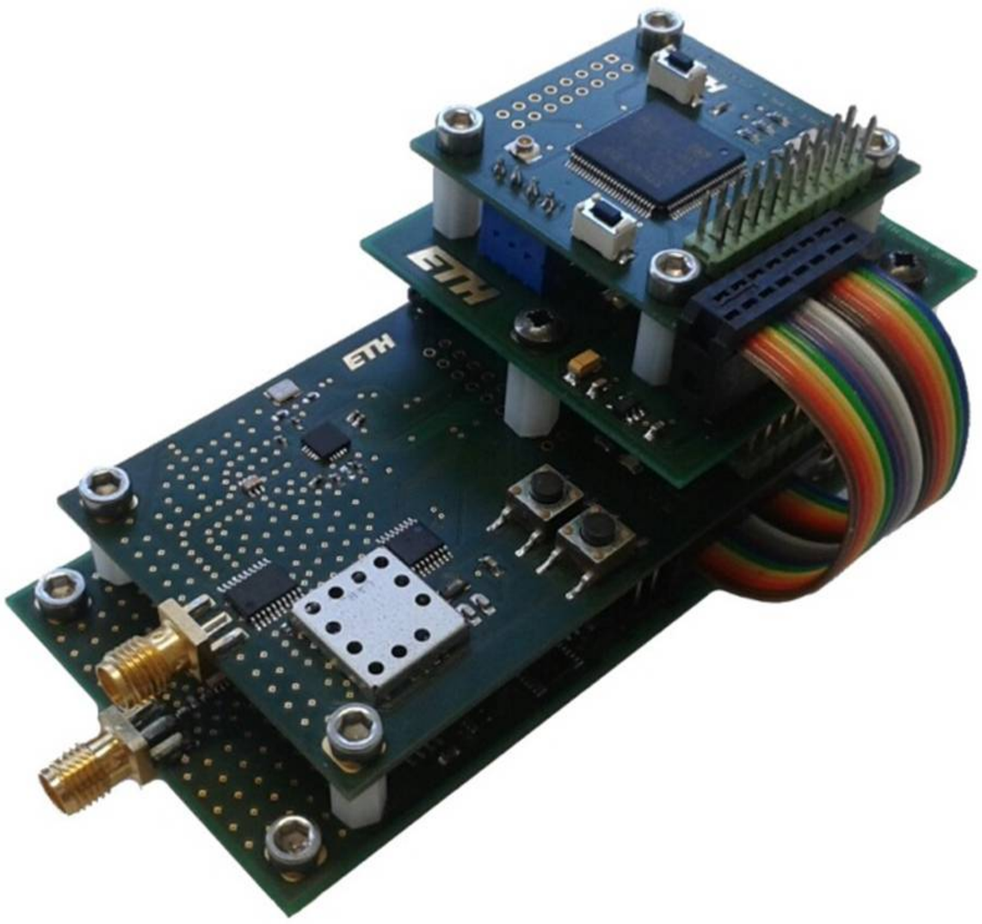
In this section we detail a prototype implementation of Blitz. In order to exemplify the use of Blitz in a real-world application scenario, we extend the design of the wireless acoustic emission sensing system presented in Chapter 3 to incorporate the Blitz dissemination of acoustic events and their characterization.

By utilizing the event-triggered design methodology presented in Chapter 3, we extend the sensing system's pipeline of event-triggered components to include the functional behavior of Blitz, as illustrated in Figure 5.17. We represent the Blitz functionality using a *(i) wake-up classification* component that implements the decision tree classifier as presented in Section 5.3, a *(ii) wake-up preamble* component for the transmission of a wake-up preamble using carrier frequency randomization as introduced in Chapter 4, and a *(iii) synchronous dissemination* component which incorporates the SYNC, EVENT and DATA rounds of the eLWB protocol as detailed in Chapter 3.

We then map the logical event-triggered components on to a physical architecture, as depicted in Figure 5.18(a). We map the wake-up



(a)



(b)

Figure 5.18: (a) Physical platform architecture, and (b) prototype of the wireless embedded platform supporting Blitz consisting of (bottom-to-top) the DPP, Zippy platform, ASI and SigGen.

classification and wake-up preamble components to the Zippy platform, as this integrates the necessary ultra-low power OOK receiver and an FSK transmitter for generating a wake-up preamble using carrier frequency randomization. The synchronous dissemination component is mapped onto the communication processor of the Dual-Processor Platform (DPP). The acoustic event characterization component used by the source nodes, and the acoustic event analysis component used by the host are mapped onto the DPP application processor. In order to trigger the prototype with reproducible acoustic events, we substitute the piezoelectric acoustic sensor with a custom-build signal generator (SigGen), which is programmed with an acoustic signal extracted from the field. The fully-integrated wireless embedded platform used for the evaluation of Blitz is illustrated in Figure 5.18(b).

5.6 Experimental Evaluation

We next experimentally evaluate the performance of Blitz with respect to latency in an indoor testbed and energy efficiency in a laboratory setting.

5.6.1 Blitz Latency

Setup. We deployed 11 Blitz-compatible nodes according to the FlockLab deployment map illustrated in Figure 5.6(a). Node 16 was configured as the host, while node 1 was configured to initiate periodic Blitz disseminations consisting of an 8-bit event packet and its associated data packet of 32 bytes, i.e., 25 bytes of characterization data with a 7 byte header. The relay nodes 19, 22, and 33 were only capable of wake-up classification and wake-up preamble transmission, and were not controlled by the FlockLab infrastructure. All nodes were configured with the wake-up classifier configuration evaluated in Section 5.3.4 with the number of wake-up preamble transmissions set to $N = 1$. A total of 100 events were generated at node 1 with $T_{event} = 5\,\text{s}$. Once the host receives an event and its associated data, it schedules a second SYNC round that contains a broadcast message informing all source nodes to return to the deep sleep listening state.

We evaluate the performance of Blitz using the following metrics: *(i) wake-up reception rate (WRR)* is the ratio of the number of nodes that wake-up compared to the number of events generated, *(ii) wake-up delay* is the time between an event generated at node 1 and the reception of the wake-up preamble, and *(iii) mean latency* is the average time between an event generated at node 1 and received at the host, and similarly, the average time between the host initiating a broadcast and when it is received by a source node.

Results. The results of the testbed evaluation are shown in Figure 5.19. The wake-up reception rate was 100% for nodes 2, 4 and 8, while the remaining nodes were above 90%. Due to limitations in the observability of GPIO pins on the relay nodes, we were unable to determine if the low

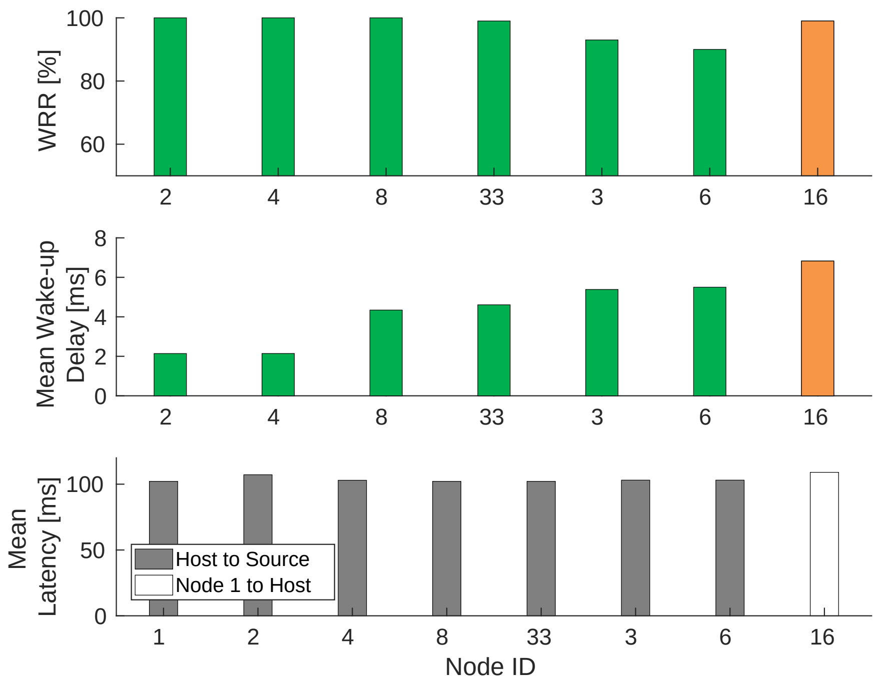


Figure 5.19: Results from the FlockLab experiment with node 16 as the host.

WRR was attributed to poor RF propagation, or the misclassification of a correct wake-up. The host failed to wake-up once during the test, thus yielding a $P_{success} = 99\%$, however whenever it did awake, it received the event and associated data without corruption. As shown in Section 5.3.4, the wake-up reliability could be improved by increasing the number of wake-up preamble retransmissions N. The wake-up delay was measured between 2.1 ms and 6.8 ms, which is consistent with the parameterization of the wake-up receiver, and the latency associated with network flooding, i.e., wake-up delay increases with hop count. The mean latency from node 1 to the host is 108.9 ms. This is very close to the latency of 109.3 ms as determined by the analytical model presented in Section 5.4 when parameterized with $h = 4$ hops. The mean latency from the host to source averaged across all nodes is 103.2 ms. We therefore conclude that the Blitz prototype achieves a low latency that is consistent with the analytical model presented in Section 5.4.

We now compare the latency of Blitz with the closest work to us in the literature incorporating a wake-up receiver, the ROD-SAN [YAE+15] radio-on-demand architecture, where a wake-up delay of 300 ms and a minimum dissemination delay of 600 ms per hop are reported. While the experimental setup differs significantly in terms of communication architecture, platform design and deployment conditions, we highlight that Blitz supports a wake-up delay on the order of milliseconds and a mean latency of 108.9 ms through a 4-hop network.

In order to put the latency of Blitz into perspective of state-of-the-art IEEE 802.15.4-based protocols, we next compare Blitz to two protocols based on Time-Slotted Channel Hopping (TSCH), as defined in the IEEE 802.15.4e standard [tsc]. Specifically, we consider the SmartMesh IP [WDSP13], a commercial protocol supporting TSCH and 6LoWPAN, and Orchestra [DANLW15], an open-source TSCH-based protocol featuring a distributed and autonomous scheduler. As reported in [WWDS15], the simulation of a low-latency configuration of SmartMesh IP exhibits an average current drain of 953 μA with a latency of 30 ms per hop, assuming nodes participate in routing and disseminate one event containing 90 bytes of payload every 30 s. Under the same conditions, Blitz would consume on average 497 μJ (139 ms @ P_{avg} = 137.9 μW, with $N = 1$ and $P_s = 16\,\mu$W), which at 2.5 V is an average current drain of approximately 6.6 μA. Therefore, assuming a 4-hop network, SmartMesh IP achieves a slightly lower latency than Blitz, however its average current drain is more than two orders of magnitude higher than Blitz. We note that platform supporting SmartMesh IP as detailed in [WDSP13], i.e., the LTC5800 system on chip, features a current drain in receive and transmit modes at least three times lower than the system on chip employed in the Blitz prototype. Alternatively, Orchestra [DANLW15] achieves approximately 400 ms latency while exhibiting an approximate 3% duty cycle in a large indoor testbed deployment consisting of 98 nodes arranged in a multi-hop network with an average of 4.2 hops. Assuming an event containing 16 bytes of payload is disseminated every 60 s through a 4-hop network, Blitz achieves a latency of approximately 109 ms with an approximate duty cycle of 0.18%. In this configuration, Blitz exhibits a latency approximately one quarter lower than Orchestra, while achieving more than one order of magnitude lower duty cycle compared to Orchestra. We acknowledge that it is very difficult to fairly compare Blitz to these SmartMesh IP and Orchestra experimental evaluations due to significant differences in protocol functionality, protocol configuration, deployment configuration, and evaluation conditions. Nevertheless, we conclude that the performance of Blitz is competitive to the aforementioned protocols.

5.6.2 Blitz Energy Efficiency

Setup. Using the RocketLogger [SGL+17] precision measurement device, we measured the power profile of Blitz source and host nodes during the dissemination of an event and its associated data. According to the experimental setup illustrated in Figure 5.20, the signal generator injected an acoustic signal into the acoustic sensor interface of the source node. The source node detected the event, characterized the acoustic signal by

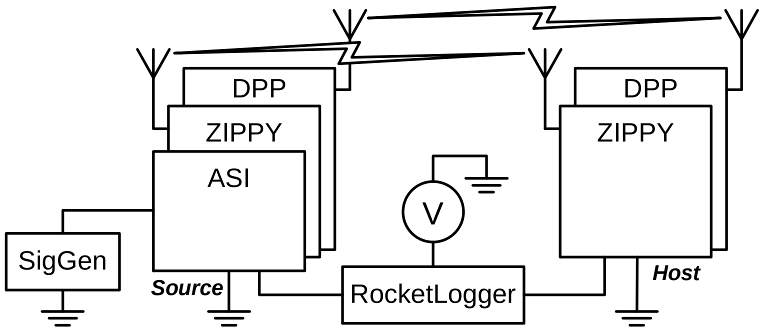


Figure 5.20: Experimental setup for measuring the power profile of Blitz source and host nodes.

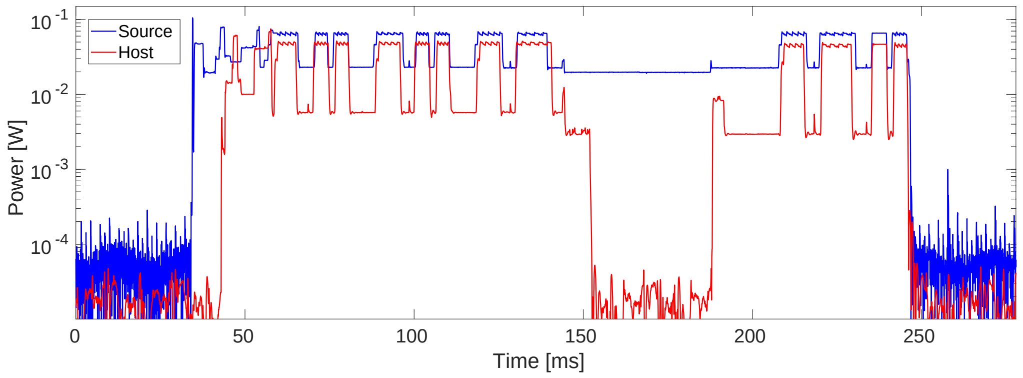


Figure 5.21: Power profile of source and host nodes using Blitz. The source node detects an acoustic event, characterises the event, and disseminates the event and its associated data to the host using Blitz.

extracting several features, before commencing a Blitz dissemination to the host located one-hop away. The source and host nodes were supplied with 2.5 V while having their on-board low-dropout regulator bypassed.

Results. As illustrated in Figure 5.21, the host node dissipated approximately 16 μW during periods of inactivity, while the source node dissipated approximately 3 μW more due to the inclusion of the acoustic sensor interface (v1.5). During the asynchronous wake-up and synchronous dissemination primitives of Blitz, the host dissipates approximately 50 mW with a transmission power of −10 dBm, while the source node dissipates 65 mW with a transmission power of 0 dBm. We highlight that not only is the source node energy-efficient by design, but also the host does not consume precious energy resources during periods of inactivity. In summary, assuming a sporadic event arrival with $T_{event} = 120$ s, the presented prototypes would operate for at least one year on a low-capacity coin cell battery.

5.7 Related Work

As surveyed in [HSR16], several protocols have been proposed for event-triggered wireless communication including T-MAC [VDL03], SIFT [JBT06] and Alert [NK08]. However, since these works are based on variants of periodic communication protocols, they all exhibit a fundamental trade-off between latency and energy efficiency. To the best of the authors knowledge, we are the first to present an architecture for event-triggered communication using interference-based flooding techniques that simultaneously support low latency and energy efficiency.

Since the introduction of wake-up receivers, several wake-up receiver-based protocols have been proposed in the literature, as surveyed in [PMK+17, DD17]. The latency and energy benefits of using wake-up receivers have been well investigated in the context of data collection in a star topology, such as in WoR-MAC [GBEM+12], WhMAC [MP12], AWD-MAC [LPM15], and SNW-MAC [AGM+16]. However, we extend these works by applying wake-up receivers to achieve multi-hop communication without having to sacrifice latency or energy efficiency.

The benefits of duty-cycling wake-up receivers have been investigated in Miller et al. [MV05], DCW-MAC [ME11, ME17], and TI-WuR/RI-WuR [ODC+16]. While duty-cycling the wake-up receiver leads to greater energy efficiency, this comes at the cost of an undesirable increase in latency, which is further exacerbated in multi-hop networks.

Only a few wake-up receiver-based protocols in the literature explicitly consider the wake-up receiver to be always-on for multi-hop communication, most notably, WUR-MAC [MD09], FLOOD-WUP [PSTT14], GWR-MAC [KPI+14], SCM-WuR [ODC+13b], and W-MAC [PIM17]. However, these proposals make simplifying assumptions on network topology and arbitration of network bandwidth, and lack experimental evaluation in a multi-hop network deployment. Specifically, line or binary tree topologies are typically assumed, which circumvents the challenges associated with the rapid and robust wake-up of realistic multi-hop topologies. Furthermore, the allocation of network bandwidth is typically contention-based, whereby a random back-off mechanism exacerbates latency and severely hinders energy efficiency. We instead tackle these challenges with Blitz, and experimentally evaluate a prototype implementation in an indoor testbed.

Raza et al. [RBF+16] propose a novel approach for efficient multi-hop data collection using model-based sensing (MBS) in conjunction with asynchronous rendezvous facilitated by wake-up receivers. If one was to place low latency constraints on the data dissemination, Blitz could be extended with an MBS scheme to reduce the number of events disseminated, and further improving energy efficiency.

The protocol partitioning between Blitz and the eLWB presented in this chapter extends and refines existing analytical studies on the benefits of wake-up receiver-based protocols over duty-cycled protocols as presented in [LMB+09, ZHDdG09, SWP10, SMB+15, ODC+16, AGM+17]. We compare Blitz to a synchronous protocol for multi-hop dissemination under sporadic events, as opposed to deterministic periodic event arrivals, and highlight the superiority of Blitz in terms of energy efficiency compared to a state-of-the-art synchronous protocol when a key design constraint is to minimize worst-case latency.

Wireless communication technologies such as backscatter [LPT+13, KPG+15] and RFID [JRO10] are well suited to a range of wireless sensing applications. However, the limited range and reliance on the existence of high-powered RF signals to facilitate communication limits their adoption, particularly when high spatial-diversity and per-hop link ranges greater than ten meters are desired.

5.8 Summary

This chapter presented the design, analysis, and experimental evaluation of Blitz, a communication architecture for efficient event-triggered multi-hop wireless communication. Blitz utilizes interference-based flooding whereby all nodes participate in the dissemination of an event and its associated data using network flooding, while embracing the interference caused by simultaneous transmissions. Blitz incorporates a novel scheme for mitigating erroneous wake-ups, which has been shown to reduce energy consumption during an indoor testbed experiment. Using an analytical model, we compared Blitz with a state-of-the-art periodic communication protocol and found Blitz to be superior in terms of latency and energy consumption per event for a wide range of event-triggered application scenarios. Finally, we experimentally evaluated a prototype implementation of Blitz which exhibits a mean latency of 108.9 ms across 4-hops, while dissipating only 16 μW during periods of inactivity.

6 Conclusions and Outlook

Cyber-physical systems are a key technology enabler for the Industrial Internet of Things, where cyber technologies providing computation and communication are used to monitor and control physical processes. Given the unpredictable nature of many real-world processes, cyber-physical systems must rapidly detect, characterize and react to non-deterministic events produced by the physical process under observation. Furthermore, in order to achieve long-term operation and to cope with the inherent heterogeneity of the physical process, these event-triggered systems must also consume energy sparingly and be adaptable to changes in the environment.

In practice, it is difficult to realize such efficient event-triggered cyber-physical systems due to three main challenges, namely, *(i)* the existence of resource interference on modern wireless embedded platforms, *(ii)* the need for quantifying component-level trade-offs despite high system complexity, and *(iii)* the fundamental trade-off between latency and energy-efficiency for multi-hop communication.

Firstly, increased processing demands on computation coupled with stringent real-time constraints on communication lead to resource interference on modern single and multi-processor wireless embedded platforms. The resource interference associated with the interleaved or concurrent execution of computation and communication tasks adversely impacts the performance of the platform in terms of responsiveness and energy-efficiency.

Secondly, as these systems are built from a complex combination of analog, digital and software components, it is difficult for the system designer to assess dependencies and quantify how component-level trade-offs impact system-level performance. However, the system designer must make component-level trade-offs in order to realize the conflicting system requirements of responsiveness, energy-efficiency and adaptability.

Thirdly, modern multi-hop wireless protocols based on periodic communication suffer from a fundamental trade-off between latency and energy efficiency. That is, to reduce the latency of multi-hop event dissemination, all nodes in the network must increase the periodicity of communication. This leads to poor energy-efficiency as frequent periodic communication is performed irrespective of whether there is an event to disseminate or not.

6.1 Contributions

To address the aforementioned challenges, we have made four main contributions in this thesis.

Composable and Energy-efficient Platform Architecture. We propose a new platform architecture where computation and communication are mapped onto dedicated processors. The information transfer between the processors is then facilitated by Bolt, a processor interconnect supporting asynchronous message passing with predictable run-time behavior and negligible power overhead. Since the worst-case execution time of Bolt message operations are tightly bounded, Bolt facilitates the composable construction of heterogeneous wireless embedded platforms, and thereby promotes independent design and component-level re-use. The benefits of Bolt are demonstrated through a prototype heterogeneous platform featuring timing-predictable inter-processor message operations with a throughput of up to 3.3 Mbps, and a power overhead of 1.3 μW during periods of inactivity.

Design Methodology for Efficient Event-triggered Cyber-physical Systems. We manage the inherent complexity and quantify component-level trade-offs by utilizing a combination of well-established design principles together with a design guideline. A four step design process is proposed whereby an event-triggered cyber-physical system is partitioned into a pipeline of modular components that adhere to an event-triggered interface. An abstract representation of each component is used to quantify responsiveness, energy-efficiency and adaptability design constraints. All components are then designed and implemented in isolation using an appropriate design guideline, before they are integrated together on a physical platform architecture.

The proposed design methodology is exemplified with the design and implementation of a prototype wireless acoustic emission sensing system, consisting of an acoustic sensor interface, an event characterization component and an event-based synchronous protocol termed the eLWB for multi-hop event dissemination. An experimental evaluation shows significant improvements to the state-of-the-art with respect to the responsive and energy-efficient detection and the multi-hop dissemination of acoustic events. Specifically, the prototype system detects an acoustic event with a delay of 16 μs, starts to characterize an acoustic event with a delay of 29 μs, disseminates an acoustic event and its associated data over a multi-hop network with a best-case latency of 113.2 ms, while dissipating a total of 59.7 μW during periods of inactivity.

On-demand Network Flooding. We demonstrate the first step toward circumventing the fundamental trade-off between latency and energy-efficiency exhibited by modern radio duty-cycled multi-hop protocols. We utilize the ultra-low power dissipation and the unique timing properties of low-complexity radios to construct Zippy, an asynchronous flooding primitive for the multi-hop dissemination of rare events. A prototype implementation of Zippy demonstrates asynchronous network wake-up, fine-grained per-hop synchronization, and efficient multi-hop dissemination of a small event packet to the host. Experiments show a best-case latency of 24.4 ms for an 8-bit event packet through a 3-hop network, a per-hop synchronization as low as 21.9 μs, and a power dissipation of 9.6 μW during periods of inactivity. Zippy therefore enables responsive dissemination of rare events while supporting multi-year operation using low-capacity coin cell batteries.

Low-latency and Energy-efficient Communication Architecture. We present Blitz, the first communication architecture that combines the rapid asynchronous network wake-up provided by Zippy, with the synchronous multi-hop dissemination of the eLWB, to achieve unprecedented latency and energy-efficiency. Specifically, Blitz combines the use of ultra-low power wake-up receivers to wake-up a network on-demand with the topology-agnostic synchronization and adaptable bandwidth allocation of the eLWB to achieve rapid dissemination of events and their associated data.

Blitz integrates a novel classification technique for mitigating erroneous wake-ups, a fundamental limitation of ultra-low power wake-up receivers that adversely impacts energy-efficiency. After collecting ground truth and identifying dominant features from extensive correct and erroneous wake-up signal traces, a wake-up classifier is designed to distinguish between correct and erroneous wake-ups. Using an analytical model, a power-optimal configuration of the wake-up classifier is determined. Experiments in an indoor testbed demonstrate the desired functionality of the wake-up classifier, which results in significant energy savings that prolong system operation.

Using an analytical model, we show that Blitz supports a lower worst-case latency and lower energy per event compared to the eLWB for a wide-range of event-triggered application scenarios. The analysis not only highlights the superiority of Blitz over the eLWB for rare events, but also for frequently occurring events that have low-latency constraints.

A prototype of Blitz is presented and incorporated into the existing wireless acoustic emission sensing system prototype. Experiments show that the developed prototype supports a best-case latency of 108.9 ms for an 8-bit event packet and its associated data packet of 32 bytes through a 4-hop network, while dissipating 16 μW, plus an additional 3 μW for the acoustic sensor interface, during periods of inactivity.

We conclude that the Blitz communication architecture is at least 5× faster than Zippy at disseminating an 8-bit event and 32 bytes of data, while supporting the codetection of both rare and frequent events. Compared to the evaluated eLWB implementation, the latency of Blitz is up to two orders of magnitude faster, and dissipates at least 3× less power during periods of inactivity.

6.2 Possible Future Directions

The contributions of this thesis represent an important step towards the construction of efficient event-triggered cyber-physical systems. We envision that the platform and communication architecture presented in this work can be further improved upon and extended. The following list summarizes possible directions for future research.

Multi-processor Interconnect. Current research trends in edge computing and transient computing are advocating an increase of computational resources to the nodes at the edge of the network, while placing the edge nodes under extreme energy constraints. This necessitates interconnecting more than two processing elements on a transiently powered wireless embedded platform. Extending the Bolt processor interconnect to support a many-processor interconnect would introduce new challenges with respect to predictable platform execution, scheduling and power management. Specifically, novel hardware and software co-design is needed to support the execution of many concurrent message controllers with predictable run-time behavior, the analysis and development of scheduling algorithms is required to support multiple message flows with priorities, and new approaches to adaptive power management are essential for the efficient management of intermittent energy sources.

Network Scalability. The network bandwidth consumed by Blitz to arbitrate network resources for event dissemination scales linearly with the number of nodes in the network. While this mechanism is feasible for small networks, it does not scale well for large networks. However, a hierarchical network may be used to support large deployments, where multiple small networks operate concurrently using orthogonal frequency domains and possibly utilizing other communication primitives. For example, the all-to-all communication primitive, such as the one presented in Chaos [LFZ13a], could be used to determine the number of events to disseminate and the corresponding bandwidth requirements. Depending on the size of the network, the time to converge for an all-to-all communication primitive may be significantly shorter than the duration of the static mechanism employed by Blitz. Future research may investigate how an event-triggered communication architecture may function efficiently across several hierarchical layers without adversely impacting responsiveness, energy-efficiency and adaptability.

Heterogeneous Protocols. The Blitz communication architecture is based on the combination of asynchronous and synchronous primitives, and has been shown to achieve significant improvements in terms of latency and energy-efficiency for short-range wireless scenarios. However, with the emergence of low-power wide-area network technologies (LPWAN), as surveyed in [RKS17], a future research direction is to investigate if the heterogeneous communication primitives employed in Blitz can deliver similar enhancements to long-range wireless scenarios. Another research direction may consider if heterogeneous communication primitives bring significant benefits when short-range and long-range wireless technologies are combined on a single wireless embedded platform.

The recent standardization efforts by the IEEE 802.11ba Task Group propose a Low-Power Wake-Up Radio (LP-WUR) for IEEE 802.11 Wireless LAN networks [McC17]. If this initiative is approved, it will likely encourage the commercialization of advanced wake-up receiver hardware featuring a power dissipation of less than $100\,\mu$W and a receiver sensitivity of at least -82 dBm. Compared to the ultra-low power receiver used in Zippy and Blitz, the standardized hardware would exhibit at least 1.5× higher receiver sensitivity with a justifiable increase in power dissipation. Commercial availability of this hardware would significantly reduce the complexities of designing and prototyping custom ultra-low power radios, and therefore foster the development of novel heterogeneous protocols for the Industrial Internet of Things.

Bibliography

[ABC$^+$04] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru, and A. Wood. EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 582–589, 2004.

[ABD15] Y. Ammar, S. Bdiri, and F. Derbel. An ultra-low power wake up receiver with flip flops based address decoder. In *Proceedings of the 12th International Multi-Conference on Systems, Signals & Devices*, pages 1–5. IEEE, 2015.

[AC14] M. P. Andersen and D. E. Culler. System design trade-offs in a next-generation embedded wireless platform. Technical Report Technical Report No. UCB/EECS-2014-162, University of California at Berkeley, 2014.

[ACDB15] U. Antao, J. Choma, A. Dibazar, and T. Berger. 40nw subthreshold event detector chip for seismic sensors. In *IEEE International Symposium on Technologies for Homeland Security*, pages 1–6. IEEE, 2015.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFC16] M. P. Andersen, G. Fierro, and D. E. Culler. System design for a synergistic, low power mote/BLE embedded platform. In *15th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 1–12. IEEE, 2016.

[AGM$^+$16] F. A. Aoudia, M. Gautier, M. Magno, O. Berder, and L. Benini. SNW-MAC: An Asynchronous Protocol Leveraging Wake-Up Receivers for Data Gathering in Star Networks. In *International Conference on Sensor Systems and Software*, pages 3–14. Springer, 2016.

[AGM$^+$17] F. A. Aoudia, M. Gautier, M. Magno, O. Berder, and L. Benini. A generic framework for modeling mac protocols in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 25(3):1489–1500, 2017.

[Amm14] H. Ammari. *The Art of Wireless Sensor Networks, Volume 1: Fundamentals*. Springer, 2014.

[APM09] J. Ansari, D. Pankin, and P. Mähönen. Radio-triggered wake-ups with addressing capabilities for extremely low power sensor network applications. *International Journal of Wireless Information Networks*, 2009.

[ARM] ARM. Cortex-M Series. `http://www.arm.com/products/processors/cortex-m/`.

[Ash92] D. Ash. A comparison between OOK/ASK and FSK modulation techniques for radio links. Technical report, RF Monolithics Inc, 1992.

[BBF$^+$11] J. Beutel, B. Buchli, F. Ferrari, M. Keller, and M. Zimmerling. X-sense: Sensing in extreme environments. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2011.

[BDH$^+$04] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald. Next-generation prototyping of sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 291–292. ACM, 2004.

[BDH10] H. Ba, I. Demirkol, and W. Heinzelman. Feasibility and benefits of passive RFID wake-up radios for wireless sensor networks. In *Global Telecommunications Conference*. IEEE, 2010.

[Beu06] J. Beutel. Fast-prototyping using the BTnode platform. In *Proceedings of the Conference on Design, Automation and Test in Europe*, volume 1, pages 1–6. IEEE, 2006.

[BFE$^+$07] J. Brown, J. Finney, C. Efstratiou, B. Green, N. Davies, M. Lowton, and G. Kortuem. Network interrupts: supporting delay sensitive applications in low power wireless control networks. In *Proceedings of the 2nd ACM Workshop on Challenged Networks*, pages 51–58. ACM, 2007.

[BJKK15] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti. BackFi: High Throughput WiFi Backscatter. *ACM SIGCOMM Computer Communication Review*, 45(4):283–296, 2015.

[BK08] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press Cambridge, 2008.

[BKK15] J. Blanckenstein, J. Klaue, and H. Karl. A survey of low-power transceivers and their applications. *IEEE Circuits and Systems Magazine*, 15(3):6–17, 2015.

[BP05] A. Y. Benbasat and J. A. Paradiso. A compact modular wireless sensor platform. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. IEEE Press, 2005.

[BS10] O. Berder and O. Sentieys. PowWow: Power optimized hardware/software framework for wireless motes. In *23rd International Conference on Architecture of Computing Systems*, pages 1–5. VDE, 2010.

[BYAH06] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pages 307–320. ACM, 2006.

[CEP16] F. Chraim, Y. B. Erol, and K. Pister. Wireless gas leak detection and localization. *IEEE Transactions on Industrial Informatics*, 12(2):768–779, 2016.

[Cha84] D. M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, 1984.

[CL00] F. Cassez and K. Larsen. The impressive power of stopwatches. In *International Conference on Concurrency Theory*, pages 138–152. Springer, 2000.

[CLY$^+$12] J. Choi, K. Lee, S.-O. Yun, S.-G. Lee, and J. Ko. An interference-aware 5.8GHz wake-up radio for ETCS. In *IEEE International Solid-State Circuits Conference*, pages 446–448, 2012.

[CLZ$^+$16] M. Cattani, A. Loukas, M. Zimmerling, M. Zuniga, and K. Langendoen. Staffetta: Smart Duty-Cycling for Opportunistic Data Collection. In *Proceedings of the 14th ACM Conference on Embedded Networked Sensor Systems*, pages 56–69, 2016.

[CRM$^+$08] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti, and R. Kumar. Brimon: A sensor network system for railway bridge monitoring. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pages 2–14. ACM, 2008.

[Cro] Crossbow Technology. `http://www.xbow.com`.

[CWA$^+$10] Q. Cao, D. Wang, T. Abdelzaher, B. Priyantha, J. Liu, and F. Zhao. Energy-optimal batching periods for asynchronous multistage data processing on sensor nodes: Foundations and an mPlatform case study. In *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010.

[DANLW15] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: Robust mesh networks through autonomously scheduled TSCH. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 337–350. ACM, 2015.

[DC08] P. Dutta and D. Culler. Epic: An open mote platform for application-driven design. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, pages 547–548. IEEE, 2008.

[DCL13] M. Doddavenkatappa, M. C. Chan, and B. Leong. Splash: Fast data dissemination with constructive interference in wireless sensor networks. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation*, pages 269–282, 2013.

[DD17] F. Z. Djiroun and D. Djenouri. MAC Protocols With Wake-Up Radio for Wireless Sensor Networks: A Review. *IEEE Communications Surveys & Tutorials*, 19(1):587–618, 2017.

[DDHC+10] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM, 2010.

[DDHC+12] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. A-MAC: A versatile and efficient receiver-initiated link layer for low-power wireless. *ACM Transactions on Sensor Networks*, 8(4):30, 2012.

[DEA06] I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: A survey. *IEEE Communications Magazine*, 2006.

[DEO09] I. Demirkol, C. Ersoy, and E. Onur. Wake-up receivers for wireless sensor networks: Benefits and challenges. *IEEE Wireless Communications*, 2009.

[DFFMM06] H. Dubois-Ferrière, L. Fabre, R. Meier, and P. Metrailler. TinyNode: A comprehensive platform for wireless sensor network applications. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 358–365. ACM, 2006.

[dFZ11] R. de Francisco and Y. Zhang. An interference robust multi-carrier wake-up radio. In *IEEE Wireless Communications and Networking Conference*, pages 1265–1270. IEEE, 2011.

[DGA+05] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *Proceedings of the 4th International*

Symposium on Information Processing in Sensor Networks. IEEE, 2005.

[DGV04] A. Dunkels, B. Grönvall, and T. Voigt. Contiki – A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462. IEEE, 2004.

[DHS12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2012.

[DLGC15] J. Dong, E. Lowenhar, V. Godinez, and M. Carlos. State-of-the-art wireless acoustic emission system for structural health monitoring. In *Advances in Acoustic Emission Technology*, pages 15–22. Springer, 2015.

[DLZL15] W. Du, J. C. Liando, H. Zhang, and M. Li. When pipelines meet fountain: Fast data dissemination in wireless sensor networks. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 365–378. ACM, 2015.

[DPZMW^{+}16] M. Delgado Prieto, D. Zurita Millan, W. Wang, A. Machado Ortiz, J. A. Ortega Redondo, and L. Romeral Martinez. Self-powered wireless sensor applied to gear diagnosis based on acoustic emission. *IEEE Transactions on Instrumentation and Measurement*, 65(1):15–24, 2016.

[DSZ^{+}16] N. Dziengel, M. Seiffert, M. Ziegert, S. Adler, S. Pfeiffer, and J. Schiller. Deployment and evaluation of a fully applicable distributed event detection system in wireless sensor networks. *Ad Hoc Networks*, 37:160–182, 2016.

[EDR^{+}11] M. Ervasti, S. Dashti, J. Reilly, J. D. Bray, A. Bayen, and S. Glaser. iShake: Mobile phones as seismic sensors–user study findings. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, pages 43–52. ACM, 2011.

[EHD04] A. El-Hoiydi and J.-D. Decotignie. WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. In *Proceedings of the 9th International Symposium on Computers and Communications*, volume 1, pages 244–251. IEEE, 2004.

[EK16] A. Engel and A. Koch. Heterogeneous wireless sensor nodes that target the Internet of Things. *IEEE Micro*, 36(6):8–15, 2016.

[FPFP16] G. Feltrin, N. Popovic, K. Flouri, and P. Pietrzak. A wireless sensor network with enhanced power efficiency and embedded strain cycle identification for fatigue monitoring of railway bridges. *Journal of Sensors*, 501:4359415, 2016.

[Fre] Freescale. VF3xxR and MKW2xDx Series. `http://www.freescale.com`.

[FZMT12] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 1–14. ACM, 2012.

[FZTS11] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks*. IEEE, 2011.

[GAJ$^+$04] D. H. Goldberg, A. G. Andreou, P. Julian, P. O. Pouliquen, L. Riddle, and R. Rosasco. A wake-up detector for an acoustic surveillance sensor network: Algorithm and VLSI implementation. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 134–141. ACM, 2004.

[GBEM$^+$12] D. Gordon, M. Berning, R. El Masri, M. Beigl, J. Blanckenstein, and J. Klaue. WoR-MAC: Combining wake-on-radio with quality-of-service for intelligent environments. In *Ninth International Conference on Networked Sensing Systems*, pages 1–8. IEEE, 2012.

[GBG$^+$12] L. Girard, J. Beutel, S. Gruber, J. Hunziker, R. Lim, and S. Weber. A custom acoustic emission monitoring system for harsh environments: application to freezing-induced damage in alpine rock walls. *Geoscientific Instrumentation, Methods and Data Systems*, 1(2):155–167, 2012.

[GKSR12] G. Gamm, M. Kostic, M. Sippel, and L. Reindl. Low-power sensor node with addressable wake-up on-demand capability. *International Journal of Sensor Networks*, 11(1):48–56, 2012.

[GO08] C. U. Grosse and M. Ohtsu. *Acoustic Emission Testing*. Springer Science & Business Media, 2008.

[GR12] G. U. Gamm and L. M. Reindl. Smart metering using distributed wake-up receivers. In *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*. IEEE, 2012.

[GRK10] W. Granzer, C. Reinisch, and W. Kastner. Future challenges for building automation: Wireless and security. In *Proceedings of the IEEE International Symposium on Industrial Electronics*, 2010.

[GS12] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. American Mathematical Society, 2012.

[GSKR10] G. U. Gamm, M. Sippel, M. Kostic, and L. M. Reindl. Low power wake-up receiver for wireless sensor nodes. In *Proceedings of the 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE, 2010.

[GSR14] G. Gamm, S. Stoecklin, and L. Reindl. Wake-up receiver operating at 433 MHz. In *Proceedings of the 11th International Multi-Conference on Systems, Signals and Devices*, 2014.

[GZR01] C. Guo, L. C. Zhong, and J. M. Rabaey. Low Power Distributed MAC for Ad Hoc Sensor Radio Networks. In *IEEE Global Telecommunications Conference*. IEEE, 2001.

[har] HART communication foundation. `https://fieldcommgroup.org/hart-specifications`.

[HC02] J. L. Hill and D. E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[HGBH09] A. Hansson, K. Goossens, M. Bekooij, and J. Huisken. CoMPSoC: A template for composable and predictable multi-processor system on chips. *ACM Transactions on Design Automation of Electronic Systems*, 14(1):2, 2009.

[HH89] P. Horowitz and W. Hill. *The Art of Electronics*. Cambridge University Press, 1989.

[HJT12] W. Heemels, K. H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *Proceedings of the IEEE 51st Annual Conference on Decision and Control*, pages 3270–3285. IEEE, 2012.

[HKV$^+$14] F. Hutu, A. Khoumeri, G. Villemaud, J.-M. Gorce, et al. Wake-up radio architecture for home wireless networks. In *IEEE Radio Wireless Symposium*, 2014.

[HMH11] C. Hambeck, S. Mahlknecht, and T. Herndl. A 2.4 μW wake-up receiver for wireless sensor nodes with -71dBm sensitivity. In *IEEE International Symposium on Circuits and Systems*, pages 534–537. IEEE, 2011.

[HS07] T. A. Henzinger and J. Sifakis. The discipline of embedded systems design. *Computer*, 40(10), 2007.

[HSR16] D. C. Harrison, W. K. Seah, and R. Rayudu. Rare event detection and propagation in wireless sensor networks. *ACM Computing Surveys*, 48(4):58, 2016.

[HXS$^+$13] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi. The evolution of MAC protocols in wireless sensor networks: A survey. *IEEE communications surveys & tutorials*, 15(1):101–120, 2013.

[IMPR16] T. Istomin, A. L. Murphy, G. P. Picco, and U. Raza. Data Prediction + Synchronous Transmissions = Ultra-low Power Wireless Sensor Networks. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems*, pages 83–95. ACM, 2016.

[Jan06] A. Jantsch. Models of computation for networks on chip. In *Proceedings of the 6th International Conference on Application of Concurrency to System Design*, pages 165–178, 2006.

[JBA07] V. Jain, R. Biswas, and D. P. Agrawal. Energy-efficient and reliable medium access in sensor networks. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, 2007.

[JBT06] K. Jamieson, H. Balakrishnan, and Y. Tay. Sift: A MAC protocol for event-driven wireless sensor networks. In *Proceedings of the 3rd European Workshop on Wireless Sensor Networks*, pages 260–275. Springer, 2006.

[JHvdV$^+$09] M. Johnson, M. Healy, P. van de Ven, M. J. Hayes, J. Nelson, T. Newe, and E. Lewis. A comparative review of wireless sensor network mote technologies. In *IEEE Sensors*, pages 1439–1442. IEEE, 2009.

[JKD$^+$07] S. Jevtic, M. Kotowsky, R. P. Dick, P. A. Dinda, and C. Dowding. Lucid Dreaming: Reliable analog event detection for energy-constrained applications. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 350–359. ACM, 2007.

[JKK$^+$11] R. Jurdak, K. Klues, B. Kusy, C. Richter, K. Langendoen, and M. Brunig. Opal: A multiradio platform for high throughput wireless sensor networks. *IEEE Embedded Systems Letters*, 3(4):121–124, 2011.

[JPC14] M. Jiménez, R. Palomera, and I. Couvertier. *Introduction to Embedded Systems: Using Microcontrollers and the MSP430*. Springer, 2014.

[JRO08] R. Jurdak, A. G. Ruzzelli, and G. M. O'Hare. Multi-hop RFID wake-up radio: design, evaluation and energy tradeoffs. In *Proceedings of the 17th International Conference on Computer, Communications and Networks*. IEEE, 2008.

[JRO10] R. Jurdak, A. Ruzzelli, and G. O'Hare. Radio sleep mode optimization in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 2010.

[JSK+13] R. Jurdak, P. Sommer, B. Kusy, N. Kottege, C. Crossman, A. Mckeown, and D. Westcott. Camazotz: Multimodal Activity-based GPS Sampling. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, pages 67–78. ACM, 2013.

[JT03] A. Jantsch and H. Tenhunen. *Networks on chip*. Springer, 2003.

[KB03] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[KBS08] H. Kwon, V. Berisha, and A. Spanias. Real-time sensing and acoustic scene characterization for security applications. In *Proceedings of the 3rd International Symposium on Wireless Pervasive Computing*, pages 755–758. IEEE, 2008.

[KdNA+14] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar. Bounding memory interference delay in COTS-based multi-core systems. In *Proceedings of the 20th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 145–154, 2014.

[KKP99] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next Century Challenges: Mobile Networking for Smart Dust. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM Press, 1999.

[KLB+12] G. Kim, Y. Lee, S. Bang, I. Lee, Y. Kim, D. Sylvester, and D. Blaauw. A 695pW standby power optical wake-up receiver for wireless sensor nodes. In *Custom Integrated Circuits Conference*, pages 1–4. IEEE, 2012.

[Kle76] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*, volume 66. Wiley New York, 1976.

[KNP+13] O. Kotaba, J. Nowotsch, M. Paulitsch, S. M. Petters, and H. Theiling. Multicore in real-time systems–temporal isolation challenges due to shared resources. In *Proceedings of Workshop on Industry-Driven Approaches for Cost-effective Certification of Safety-Critical, Mixed-Criticality Systems*, 2013.

[Kop08] H. Kopetz. The complexity challenge in embedded system design. In *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, pages 3–12. IEEE, 2008.

[Kop11] H. Kopetz. *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media, 2011.

[KPC+07] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks*. IEEE, 2007.

[KPG+15] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall. Wi-Fi Backscatter: Internet connectivity for RF-powered devices. *ACM SIGCOMM Computer Communication Review*, 44(4):607–618, 2015.

[KPI+14] H. Karvonen, J. Petäjäjärvi, J. Iinatti, M. Hämäläinen, and C. Pomalaza-Ráez. A generic wake-up radio based MAC protocol for energy efficient short range communication. In *IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication*, pages 2173–2177. IEEE, 2014.

[KPK+14] Y.-s. Kuo, P. Pannuto, G. Kim, Z. Foo, I. Lee, B. Kempke, P. Dutta, D. Blaauw, and Y. Lee. MBus: A 17.5pJ/bit/chip portable interconnect bus for millimeter-scale sensor systems with 8nW standby power. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 1–4. IEEE, 2014.

[KRG12] B. M. Kelly, B. Rumberg, and D. W. Graham. An ultra-low-power analog memory system with an adaptive sampling rate. In *Proceedings of the 55th International Midwest Symposium on Circuits and Systems*, pages 302–305. IEEE, 2012.

[KW07] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2007.

[KW16] M. König and R. Wattenhofer. Maintaining constructive interference using well-synchronized sensor nodes. In *2016 International Conference on Distributed Computing in Sensor Systems*, pages 206–215. IEEE, 2016.

[LBT01] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media, 2001.

[LDX+10] G. Lu, D. De, M. Xu, W.-Z. Song, and J. Cao. TelosW: Enabling ultra-low power wake-on sensor network. In *Proceedings of the 7th International Conference on Networked Sensing Systems*, pages 211–218. IEEE, 2010.

[Lee08] E. A. Lee. Cyber physical systems: Design challenges. In *11th IEEE International Symposium on Object Oriented Real-time Distributed Computing*, pages 363–369. IEEE, 2008.

[LF76] K. Leentvaar and J. Flint. The capture effect in FM receivers. *IEEE Transactions on Communications*, 1976.

[LFST17] R. Lim, R. D. Forno, F. Sutton, and L. Thiele. Competition: Robust Flooding using Back-to-Back Synchronous Transmissions with Channel-Hopping. In *Proceedings of the 14th International Conference on Embedded Wireless Systems and Networks*, volume 17, pages 270–271, 2017.

[LFZ13a] O. Landsiedel, F. Ferrari, and M. Zimmerling. Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 1. ACM, 2013.

[LFZ$^+$13b] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*. ACM, 2013.

[LHV$^+$09] Á. Lédeczi, T. Hay, P. Völgyesi, D. R. Hay, A. Nádas, and S. Jayaraman. Wireless acoustic emission sensor network for structural monitoring. *IEEE Sensors Journal*, 9(11):1370–1377, 2009.

[LMB$^+$09] M. Lont, D. Milosevic, P. G. Baltus, A. H. Van Roermund, and G. Dolmans. Analytical models for the wake-up receiver power budget for wireless sensor networks. In *IEEE Global Telecommunications Conference*, pages 1–6. IEEE, 2009.

[LMD$^+$15] R. Lim, B. Maag, B. Dissler, J. Beutel, and L. Thiele. A testbed for fine-grained tracing of time sensitive behavior in wireless sensor networks. In *Local Computer Networks Workshops*, 2015.

[LMP$^+$05] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. TinyOS: An operating system for sensor networks. *Ambient intelligence*, 35:115–148, 2005.

[LMvR14] M. Lont, D. Milosevic, and A. van Roermund. *Wake-up Receiver Based Ultra-low-power WBAN*. Springer, 2014.

[LPM15] T. N. Le, A. Pegatoquet, and M. Magno. Asynchronous on demand MAC protocol using wake-up radio in wireless body area network. In *6th IEEE International Workshop on Advances in Sensors and Interfaces*, pages 228–233. IEEE, 2015.

[LPT+13] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith. Ambient backscatter: Wireless communication out of thin air. In *Proceedings of the ACM SIGCOMM Conference*. ACM, 2013.

[LPY97] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1):134–152, 1997.

[LRW04] E.-Y. Lin, J. M. Rabaey, and A. Wolisz. Power-efficient rendezvous schemes for dense wireless sensor networks. In *International Conference on Communications*. IEEE, 2004.

[LS16] E. A. Lee and S. A. Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2016.

[Mar94] G. Marsaglia. The mother of all random generators, 1994.

[McC17] D. K. McCormick. Preview: IEEE Technology Report on Wake-Up Radio. pages 1–11, 2017.

[MD09] S. Mahlknecht and M. S. Durante. WUR-MAC: Energy efficient wakeup receiver based MAC protocol. In *Proceedings of the 8th IFAC Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, 2009.

[MDN13] D. Messaoud, D. Djamel, and B. Nadjib. Survey on Latency Issues of Asynchronous MAC Protocols in Delay-Sensitive Wireless Sensor Networks. *IEEE Communications Surveys Tutorials*, 15(2):528–550, 2013.

[ME11] N. S. Mazloum and O. Edfors. DCW-MAC: An energy efficient medium access scheme using duty-cycled low-power wake-up receivers. In *IEEE Vehicular Technology Conference*, pages 1–5. IEEE, 2011.

[ME17] N. S. Mazloum and O. Edfors. Influence of Duty-Cycled Wake-Up Receiver Characteristics on Energy Consumption in Single-Hop Networks. *IEEE Transactions on Wireless Communications*, 16(6):3870–3884, 2017.

[MELT08] R. Musaloiu-E, C.-J. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. IEEE, 2008.

[MJS+16] M. Magno, V. Jelicic, B. Srbinovski, V. Bilas, E. Popovici, and L. Benini. Design, implementation, and performance evaluation of a flexible low-latency nanowatt wake-up radio receiver. *IEEE Transactions on Industrial Informatics*, 12(2):633–644, 2016.

[MMF+07] M. Malinowski, M. Moskwa, M. Feldmeier, M. Laibowitz, and J. A. Paradiso. CargoNet: A low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events. In *Proceedings of the 5th International Conference on Embedded networked Sensor Systems*, pages 145–159. ACM, 2007.

[MP12] S. Marinkovic and E. Popovici. Ultra low power signal oriented approach for wireless health monitoring. *Sensors*, 12(6):7917–7937, 2012.

[MV05] M. J. Miller and N. H. Vaidya. A MAC protocol to reduce sensor network energy consumption using a wakeup radio. *IEEE Transactions on Mobile Computing*, 4(3):228–242, 2005.

[Mye01] C. J. Myers. *Asynchronous circuit design*. John Wiley & Sons, 2001.

[NHS+08] L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling. Imote2: Serious computation at the edge. In *Wireless Communications and Mobile Computing Conference*, pages 1118–1123. IEEE, 2008.

[NK08] V. Namboodiri and A. Keshavarzian. Alert: An adaptive low-latency event-driven MAC protocol for wireless sensor networks. In *Proceedings of the 2008 International Conference on Information Processing in Sensor Networks*, pages 159–170. IEEE, 2008.

[NKA+05] L. Nachman, R. Kling, R. Adler, J. Huang, and V. Hummel. The Intel mote platform: a Bluetooth-based sensor network for industrial monitoring. In *Fourth International Symposium on Information Processing in Sensor Networks*, pages 437–442. IEEE, 2005.

[NXP] NXP Semiconductors. Cortex-M4 MCUs with Cortex-M0 Co-Processors. `http://www.nxp.com/products/microcontrollers/core/cortex_m0_m4f/`.

[OBD+05] B. O'Flynn, S. Bellis, K. Delaney, J. Barton, S. C. O'Mathuna, A. M. Barroso, J. Benson, U. Roedig, and C. Sreenan. The development of a novel minaturized modular platform for wireless sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pages 370–375. IEEE, 2005.

[OBM+16] D. Oletic, V. Bilas, M. Magno, N. Felber, and L. Benini. Low-power multichannel spectro-temporal feature extraction circuit for audio pattern wake-up. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 355–360. IEEE, 2016.

[OBR14] F. J. Oppermann, C. A. Boano, and K. Römer. A decade of wireless sensing applications: Survey and taxonomy. In *The Art of Wireless Sensor Networks*, pages 11–50. Springer, 2014.

[ODC$^+$13a] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl. Performance evaluation and comparative analysis of subcarrier modulation wake-up radio systems for energy-efficient wireless sensor networks. *Sensors*, 14(1):22–51, 2013.

[ODC$^+$13b] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl. Wake-up radio as an energy-efficient alternative to conventional wireless sensor networks MAC protocols. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, pages 173–180. ACM, 2013.

[ODC$^+$16] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl. Has time come to switch from duty-cycled MAC protocols to wake-up radio for wireless sensor networks? *IEEE/ACM Transactions on Networking*, 24(2):674–687, 2016.

[ODCP13] J. Oller, I. Demirkol, J. Casademont, and J. Paradells. Design, development, and performance evaluation of a low-cost, low-power wake-up radio system for wireless sensor networks. *ACM Transactions on Sensor Networks*, 2013.

[ODP$^+$12] J. Oller, I. Demirkol, J. Paradells, J. Casademont, and W. Heinzelman. Time-Knocking: A novel addressing mechanism for wake-up receivers. In *Proceedings of the 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 268–275. IEEE, 2012.

[Oet79] J. Oetting. A comparison of modulation techniques for digital radio. *IEEE Transactions on Communications*, 27(12):1752–1762, 1979.

[ORW13] S. Oh, N. E. Roberts, and D. D. Wentzloff. A 116nW multi-band wake-up receiver with 31-bit correlator and interference rejection. In *IEEE Custom Integrated Circuits Conference*, pages 1–4. IEEE, 2013.

[Pap65] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1965.

[PCB00] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 32–43. ACM, 2000.

[PCK16] A. Parasuram, D. Culler, and R. Katz. An analysis of the RPL routing standard for low power and lossy networks. Technical Report Technical Report No. UCB/EECS-2016-106, University of California at Berkeley, 2016.

[PDCDLTR06] J. Portilla, A. De Castro, E. De La Torre, and T. Riesgo. A modular architecture for nodes in wireless sensor networks. *Universal Computer Science*, 12(3):328–339, 2006.

[PGR07] N. Pletcher, S. Gambini, and J. Rabaey. A 65 μW, 1.9 GHz RF to digital baseband wakeup receiver for wireless sensor nodes. In *Proceedings of the Custom Integrated Circuits Conference*, pages 539–542. IEEE, 2007.

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 95–107. ACM, 2004.

[PIM17] R. Piyare, T. Istomin, and A. L. Murphy. WaCo: A Wake-Up Radio COOJA Extension for Simulating Ultra Low Power Radios. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, 2017.

[PJ09] C. V. Pollack Jr. Wireless cardiac event alert monitoring is feasible and effective in the emergency department and adjacent waiting areas. *Critical Pathways in Cardiology*, 8(1):7–11, 2009.

[PLGS15] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith. Turbocharging ambient backscatter communication. volume 44, pages 619–630. ACM, 2015.

[PMK$^+$17] R. Piyare, A. L. Murphy, C. Kiraly, P. Tosato, and D. Brunelli. Ultra Low Power Wake-Up Radios: A Hardware and Networking Survey. *IEEE Communications Surveys & Tutorials*, 19(4):2117–2157, 2017.

[PSC05] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.

[PSMJ13] T. Prabhakar, N. Soumya, P. Muralidharan, and H. Jamadagni. A Novel Wake-Up Radio WSN Mote. In *Proceedings of the Texas Instruments India Educators' Conference*, pages 362–368, 2013.

[PSTT14] C. Petrioli, D. Spenza, P. Tommasino, and A. Trifiletti. A novel wake-up receiver with addressing capability for wireless sensor nodes. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems*, pages 18–25, 2014.

[RBF$^+$16] U. Raza, A. Bogliolo, V. Freschi, E. Lattanzi, and A. L. Murphy. A two-prong approach to energy-efficient WSNs: Wake-up receivers plus dedicated, model-based sensing. *Ad Hoc Networks*, 45:1–12, 2016.

[Ree54] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 1954.

[RGK12] B. Rumberg, D. W. Graham, and V. Kulathumani. A low-power, programmable analog event detector for resource-constrained sensing systems. In *Proceedings of the 55th International Midwest Symposium on Circuits and Systems*, pages 338–341. IEEE, 2012.

[RGR07] A. Rowe, D. Goel, and R. Rajkumar. FireFly Mosaic: A vision-enabled wireless sensor networking system. In *Proceedings of the 28th IEEE International Real-time Systems Symposium*, pages 459–468. IEEE, 2007.

[RKS17] U. Raza, P. Kulkarni, and M. Sooriyabandara. Low power wide area networks: An overview. *IEEE Communications Surveys & Tutorials*, 19(2):855–873, 2017.

[RLSS10] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical Systems: The Next Computing Revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.

[ROGLA03] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 181–192. ACM, 2003.

[RWA$^+$08] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu. Z-MAC: A hybrid MAC for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 16(3):511–524, 2008.

[SBS02] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pages 160–171. ACM, 2002.

[SDGJ08] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson. DW-MAC: A low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 53–62. ACM, 2008.

[SGFCPS$^+$15] J. Segura-Garcia, S. Felici-Castell, J. J. Perez-Solano, M. Cobos, and J. M. Navarro. Low-cost alternatives for urban noise nuisance monitoring using wireless sensor networks. *IEEE Sensors Journal*, 15(2):836–844, 2015.

[SGG12] O. Stecklina, D. Genschow, and C. Goltz. TandemStack-A Flexible and Customizable Sensor Node Platform for Low Power Applications. In *1st International Conference on Sensor Networks*, pages 65–72, 2012.

[SGL$^+$17] L. Sigrist, A. Gomez, R. Lim, S. Lippuner, M. Leubin, and L. Thiele. Measurement and Validation of Energy Harvesting IoT Devices. In *Proceedings of the 2017 Design, Automation & Test in Europe Conference & Exhibition*, pages 1159–1164. IEEE, 2017.

[Sif13] J. Sifakis. Rigorous system design. *Foundations and Trends® in Electronic Design Automation*, 6(4):293–362, 2013.

[SKK$^+$12] J. Suhonen, M. Kohvakka, V. Kaseva, T. D. Hämäläinen, and M. Hännikäinen. *Low-power wireless sensor networks: protocols, services and applications*. Springer Science & Business Media, 2012.

[SLMR05] J. Stankovic, I. Lee, A. Mok, and R. Rajkumar. Opportunities and obligations for physical computing systems. *Computer*, 2005.

[SMB$^+$15] D. Spenza, M. Magno, S. Basagni, L. Benini, M. Paoli, and C. Petrioli. Beyond duty cycling: Wake-up radio with selective awakenings for long-lived wireless sensing systems. In *IEEE Conference on Computer Communications*, pages 522–530. IEEE, 2015.

[SML$^+$04] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 1–12. ACM, 2004.

[SSV12] R. K. Shakya, Y. N. Singh, and N. K. Verma. Optimizing channel access for event-driven wireless sensor networks: Analysis and enhancements. *arXiv preprint arXiv:1203.5874*, 2012.

[STGS02] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 2002.

[SURK17] J. Saez, T. Ungan, L. M. Reindl, and T. Kumberg. Development and characterization of a robust differential wake-up receiver for wireless sensor networks. In *13th International Wireless Communications and Mobile Computing Conference*, pages 1209–1214. IEEE, 2017.

[Swi16] Swiss Federal Office of Communications. Radio Interface Regulation 1021-05, 2016. `https://www.ofcomnet.ch/api/rir/1021/05`.

[SWP10] R. Su, T. Watteyne, and K. S. Pister. Comparison between preamble sampling and wake-up receivers in wireless sensor networks. In *IEEE Global Telecommunications Conference*, pages 1–5. IEEE, 2010.

[Tex] Texas Instruments. F28M3x Series. `http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/c2000_performance/control_automation/f28m3x/products.page`.

[TPM$^+$15] N. A. Tatlas, S. M. Potirakis, S. A. Mitilineos, S. Despotopoulos, D. Nicolaidis, and M. Rangoussi. A wireless acoustic sensor network for environmental monitoring based on flexible hardware nodes. In *Proceedings of the Audio Mostly 2015 on Interaction With Sound*, pages 30:1–30:5. ACM, 2015.

[TPVW16] P. Tuset-Peiró, X. Vilajosana, and T. Watteyne. OpenMote+: A range-agile multi-radio mote. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, pages 333–334, 2016.

[tsc] IEEE 802.15.4e Wireless Standard - Amendment 1: MAC sublayer. `http://standards.ieee.org/findstds/standard/802.15.4e-2012.html`.

[VBN$^+$07] P. Volgyesi, G. Balogh, A. Nadas, C. B. Nash, and A. Ledeczi. Shooter localization and weapon classification with soldier-wearable networked sensors. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, pages 113–126. ACM, 2007.

[VDL03] T. Van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 171–180. ACM, 2003.

[vHDHK03] L. van Hoesel, S. Dulman, P. Havinga, and H. Kip. Design of a low-power testbed for wireless sensor networks and verification. Technical report, University of Twente, Centre for Telematics and Information Technology (CTIT), 2003.

[VHPP$^+$17] A. Varshney, O. Harms, C. Pérez-Penichet, C. Rohner, F. Hermans, and T. Voigt. LoRea: A Backscatter Architecture that Achieves a Long Communication Range. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2017.

[VTWP15] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister. OpenMote: Open-source prototyping platform for the Industrial IoT. In *International Conference on Ad Hoc Networks*, pages 211–222. Springer, 2015.

[WAJR+05] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proceeedings of the 2nd European Workshop on Wireless Sensor Networks,*, pages 108–120. IEEE, 2005.

[WDSP13] T. Watteyne, L. Doherty, J. Simon, and K. Pister. Technical overview of SmartMesh IP. In *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 547–551. IEEE, 2013.

[WGR+09] R. Wilhelm, D. Grund, J. Reineke, M. Schlickling, M. Pister, and C. Ferdinand. Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(7):966–978, 2009.

[WLS14] M. Wilhelm, V. Lenders, and J. B. Schmitt. On the reception of concurrent transmissions in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 13(12):6756–6767, 2014.

[WSJ17] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The future of industrial communication: Automation networks in the era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017.

[WT06] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.

[WWDS15] T. Watteyne, J. Weiss, L. Doherty, and J. Simon. Industrial IEEE802.15.4e Networks: Performance and trade-offs. In *IEEE International Conference on Communications*, pages 604–609. IEEE, 2015.

[XRC+04] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 13–24. ACM, 2004.

[YAE+15] H. Yomo, K. Abe, Y. Ezure, T. Ito, A. Hasegawa, and T. Ikenaga. Radio-On-Demand Sensor and Actuator Networks (ROD-SAN): System Design and Field Trial. In *IEEE Global Communications Conference*, pages 1–6. IEEE, 2015.

[YHE02] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1567–1576. IEEE, 2002.

[YKK13] K. Yadav, I. Kymissis, and P. R. Kinget. A 4.4μW wake-up receiver using ultrasound data communications. volume 48, pages 649–660. IEEE, 2013.

[YV04] X. Yang and N. Vaidya. A wakeup scheme for sensor networks: Achieving balance between energy saving and end-to-end delay. In *Proceedings of the 10th Real-Time and Embedded Technology and Applications Symposium*, pages 19–26. IEEE, 2004.

[ZBJK16] P. Zhang, D. Bharadia, K. R. Joshi, and S. Katti. HitchHike: Practical backscatter using commodity WiFi. In *Proceedings of the 14th ACM Conference on Embedded Networked Sensor Systems*, pages 259–271, 2016.

[ZFB10] P. Zappi, E. Farella, and L. Benini. Tracking motion direction and distance with pyroelectric IR sensors. *IEEE Sensors Journal*, 10(9):1486–1494, 2010.

[ZFM$^+$12] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele. pTunes: Runtime parameter adaptation for low-power MAC protocols. In *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, pages 173–184. ACM, 2012.

[ZHDdG09] Y. Zhang, L. Huang, G. Dolmans, and H. de Groot. An analytical model for energy efficiency analysis of different wakeup radio schemes. In *IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1148–1152. IEEE, 2009.

[ZHPG14] P. Zhang, P. Hu, V. Pasikanti, and D. Ganesan. EkhoNet: High speed ultra low-power backscatter for next generation sensors. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, pages 557–568. ACM, 2014.

[Zim15] M. Zimmerling. *End-to-end Predictability and Efficiency in Low-power Wireless Networks*. PhD thesis, ETH Zurich, 2015.

[ZMK$^+$17] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele. Adaptive real-time communication for wireless cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 1(2):8, 2017.

[ZYH15] F. Zahedi, J. Yao, and H. Huang. A passive wireless ultrasound pitch–catch system. *Smart Materials and Structures*, 24(8):085030, 2015.

List of Publications

The following list of publications form the basis of this thesis. The corresponding chapters are indicated in parentheses.

F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele. **Bolt: A Stateful Processor Interconnect.** In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys).* Seoul, South Korea, November 2015. (Chapter 2)

F. Sutton, B. Buchli, J. Beutel, L. Thiele. **Zippy: On-Demand Network Flooding.** In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys).* Seoul, South Korea, November 2015. (Chapter 4)

F. Sutton, R. Da Forno, D. Gschwend, T. Gsell, R. Lim, J. Beutel, and L. Thiele. **The Design of a Responsive and Energy-efficient Event-triggered Wireless Sensing System.** In *Proceedings of the 14th International Conference on Embedded Wireless Systems and Networks (EWSN).* Uppsala, Sweden, February 2017. ***Best Paper Runner-up.*** (Chapter 3)

F. Sutton, R. Da Forno, J. Beutel, L. Thiele. **BLITZ: A Network Architecture for Low Latency and Energy-efficient Event-triggered Wireless Communication.** In *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless (HotWireless).* Snowbird, UT, USA, October 2017. (Chapter 5)

F. Sutton, J. Beutel, L. Thiele. **Poster: Mitigating Erroneous Wake-ups.** In *Proceedings of the 15th ACM Conference on Embedded Networked Sensor Systems (SenSys).* Delft, Netherlands, November 2017. ***Best Poster Award.*** (Chapter 5)

The following list includes publications that are not part of this thesis.

B. Buchli, F. Sutton, and J. Beutel. **GPS-Equipped Wireless Sensor Network Node for High-Accuracy Positioning Applications.** In *Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN).* Trento, Italy, February 2012.

F. Sutton, B. Buchli, and J. Beutel. **Demo Abstract: Ultra-Low Power Wireless Multi-Sensor Interface.** In *Proceedings of the 10th European Conference on Wireless Sensor Networks (EWSN).* Ghent, Belgium, February 2013.

M. Zimmerling, F. Ferrari, R. Lim, O. Saukh, F. Sutton, R. Da Forno, R. S. Schmidt, and M. A. Wyss. **Poster Abstract: A Reliable Wireless Nurse Call System: Overview and Pilot Results from a Summer Camp for Teenagers with Duchenne Muscular Dystrophy.** In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys).* Rome, Italy, November 2013.

B. Buchli, F. Sutton, J. Beutel, and L. Thiele. **Towards Enabling Uninterrupted Long-Term Operation of Solar Energy Harvesting Embedded Systems.** In *Proceedings of the 11th European Conference on Wireless Sensor Networks (EWSN).* Oxford, UK, February 2014.

F. Sutton, R. Da Forno, R. Lim, M. Zimmerling, and L. Thiele. **Demonstration Abstract: Automatic Speech Recognition for Resource-Constrained Embedded Systems.** In *Proceedings of the 13th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).* Berlin, Germany, April 2014.

Z. Su, A. Geiger, P. Limpach, J. Beutel, T. Gsell, B. Buchli, S. Gruber, V. Wirz, and F. Sutton. **Online Monitoring of Alpine Slope Instabilities with L1 GPS Real Time Kinematic Positions.** In *Geophysical Research Abstracts - European Geoscience Union General Assembly.* Vienna, Austria, April 2014.

S. Weber, J. Beutel, B. Buchli, S. Gruber, T. Gsell, R. Lim, P. Limpach, H. Raetzo, Z. Su, F. Sutton, C. Walser, and V. Wirz. **PermaSense L1-GPS for Kinematic Monitoring.** In *Book of Abstracts of EUCOP4 - 4th European Conference on Permafrost.* Évora, Portugal, June 2014.

B. Buchli, F. Sutton, J. Beutel, and L. Thiele. **Dynamic Power Management for Long-Term Energy Neutral Operation of Solar Energy**

Harvesting Systems. In *Proceedings of the 12th ACM Conference on Embedded Networked Sensor Systems (SenSys).* Memphis, TN, USA, November 2014.

F. Sutton, R. Da Forno, M. Zimmerling, R. Lim, T. Gsell, F. Ferrari, J. Beutel, and L. Thiele. **Poster Abstract: Predictable Wireless Embedded Platforms.** In *Proceedings of the 14th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).* Seattle, WA, USA, April 2015.

F. Sutton, L. Thiele. **Poster Abstract: Wake-up Flooding: An Asynchronous Network Flooding Primitive.** In *Proceedings of the 14th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).* Seattle, WA, USA, April 2015.

F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele. **Demo: Building Reliable Wireless Embedded Platforms using the Bolt Processor Interconnect.** In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys).* Seoul, South Korea, November 2015.

F. Sutton, R. Da Forno, D. Gschwend, R. Lim, T. Gsell, J. Beutel, and L. Thiele. **Poster Abstract: A Heterogeneous System Architecture for Event-triggered Wireless Sensing.** In *Proceedings of the 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).* Vienna, Austria, April 2016.

L. Thiele, F. Sutton, R. Jacob, R. Lim, R. da Forno, and J. Beutel. **On platforms for CPS: adaptive, predictable and efficient.** In *Proceedings of the 27th International Symposium on Rapid System Prototyping: Shortening the Path from Specification to Prototype.* Pittsburgh, Pennsylvania, October 2016.

R. Lim, R. Da Forno, F. Sutton, and L. Thiele. **Competition: Robust Flooding using Back-to-Back Synchronous Transmissions with Channel-Hopping.** In *Proceedings of the 14th International Conference on Embedded Wireless Systems and Networks (EWSN).* Uppsala, Sweden, February 2017.