# mSync: Physical Layer Frame Synchronization Without Preamble Symbols

Bastian Bloessl, *Member, IEEE*, Falko Dressler, *Fellow, IEEE*

**Abstract**—We present a novel physical layer frame format and a corresponding decoding strategy for energy-constraint single-carrier transceivers, often used in sensor networks and cyber-physical systems. The main advantage of our approach is that decoding does not rely on dedicated preamble symbols, which usually introduce considerable overhead in terms of energy consumption and utilization of the wireless channel. We show that omitting the preamble can be achieved by buffering the signal in the receiver and processing the samples twice; first to synchronize and in a second iteration to decode the actual data. To introduce our approach, we provide a theoretical description, including a discussion of the implications of synchronizing on data symbols instead of optimized preamble sequences. The practical feasibility of the algorithm is shown by simulations and experiments using prototype implementations based on software defined radio. We implemented our algorithm for two technologies, a custom ultra low-power BPSK transceiver and the O-QPSK physical layer of the IEEE 802.15.4 standard. Finally, we present an extension of the algorithm that allows us to reduce the buffered data to a small constant number of samples, making our algorithm applicable to physical layers independent from their maximum frame size.

**Index Terms**—Physical Layer, Frame Synchronization, Preamble, Low-Power Communications, Software Defined Radio

## 1 Introduction

Today, people as well as an ever increasing number of things around us are connected every time and everywhere by means of wireless communications. To support a diverse set of applications, a broad spectrum of technologies is in use. On the one end of the spectrum, there are performance-oriented technologies like LTE or WiFi that employ advanced concepts such as multi-user MIMO to boost throughput and spectral efficiency. On the other end of the spectrum, there are low-power and low-bandwidth single carrier technologies that allow for energy efficient operation. Especially with the proliferation of Industry 4.0 and the vision of an Internet of Things (IoT) [1], the second kind of devices has shifted into the focus of research and development. It is well recognized that energy efficient wireless technologies form the central building block for many applications including Wireless Sensor Networks (WSNs) [2], Cyber-Physical Systems (CPSs) [3], industrial automation, wildlife monitoring [4], [5], and medical implants. More recently, also car and plane manufacturers are looking into replacing wired system with wireless sensors to save cabling and, thus, weight and fuel. In this paper, we will use the more traditional term WSNs when referring to such energy efficient wireless communication technologies.

In WSNs, the design is mainly affected by the limited energy budget available to the device. The challenge is to find a good trade-off between functionality and performance, on the one hand, and network lifetime [6], on the other hand. To optimize energy consumption and increase network lifetime, researchers developed concepts that range from transmit power control [7], over duty-cycled MAC protocols [8], [9], to wake-up receivers [10].

- *B. Bloessl and F. Dressler are with the Heinz Nixdorf Institute and the Dept. of Computer Science, Paderborn University, Germany, E-mail: {bloessl, dressler}@ccs-labs.org*
- *B. Bloessl is also with the Connect Center, Trinity College Dublin, Ireland*

In this paper, we propose mSync, a novel orthogonal concept that can complement existing approaches. The core idea, which we briefly presented in [11], is to decode the frame without relying on dedicated preamble symbols. By avoiding the overhead of a dedicated preamble, the frames are shorter, which saves energy and reduces occupancy of the wireless channel.

The improvement can be considerable, especially since the frame sizes in WSNs are usually small. As an example, we consider the Offset Quadrature Phase-Shift Keying (O-QPSK) physical layer of the IEEE 802.15.4 standard, which forms the base of the ZigBee stack [12], [13], a popular choice for industrial automation and IoT applications. Here, the minimum preamble length is the equivalent of 4 byte compared to the total size of an Acknowledgement Frame (ACK) of 11 byte. Admittedly, an ACK presents an extreme case where our algorithm can provide a large benefit, but we also have to consider that in unicast networks they comprise 50 % of the frames. Moreover, even a maximum sized frame with 133 byte in total, the overhead of 4 byte is still non-negligible.

Apart from the O-QPSK physical layer, we apply our new concept also to a custom ultra low-power transceiver that we designed for a 2 g sensor mote, used to track bats in their natural habitat [4]. In the paper, we will refer to this custom implementation as BATS transceiver. Due to architectural constraints of the sensor mote, we are only able to send very short frames with a total length of 12 byte. In our initial design, we allocated 2 byte, i.e. over 16 %, for the preamble. This overhead can be completely avoided with mSync.

We introduce our idea by providing a thorough theoretical description that considers the implications of synchronizing on data symbols instead of optimized preamble sequences. To show the practical feasibility and the general applicability of the approach, we implement SDR-based prototypes for two very different technologies: the BATS transceiver and the

IEEE 802.15.4 O-QPSK physical layer. Using these prototypes, we assess the performance by means of simulations as well as experiments. The results highlight that our approach allows us to eliminate the overhead of preamble symbols without degrading physical layer performance. On the contrary, omitting the preamble reduces the air-time of the frame, reducing occupancy of the wireless channel. To study this effect in greater detail, we employ a Markov model of the IEEE 802.15.4 MAC to show how shorter frames can improve the saturation goodput.

However, also with mSync, there is no free lunch. The drawback of a naïve implementation is that the receiver has to buffer samples corresponding to a maximum sized frame. Depending on the technology, this could waste a lot of resources and void the advantage in terms of energy consumption. To address this issue, we present a slight variation of the algorithm that allows us to reduce the buffered data to a small number of samples independent from the maximum frame size. With this, we show that mSync is applicable to many packet-based single-carrier technologies and presents an interesting new option in the design space of low-power wireless communications.

Our main contributions can be summarized as follows:

- We present a novel frame format and a corresponding decoding strategy that allows to save the overhead of preamble symbols without degrading physical layer performance.
- We demonstrate the feasibility and general applicability of our approach by implementing SDR-based prototypes for two communication technologies: our BATS transceiver and the O-QPSK physical layer of IEEE 802.15.4.
- We investigate the reduced energy consumption, discuss the possibility to use more robust receiver configurations, and show how shorter frames (i.e., frames without a preamble) can considerably increase system level goodput.
- We finally show that, with a small variation of the algorithm, the amount of data that has to be buffered can be reduced to a small fixed number of samples, making our approach applicable also to physical layers with large maximum frame sizes.

## 2 RELATED WORK

Driven by the idea of *smart dust* [14], many researchers began working towards distributed, decentralized, and self-organizing networks of small sensor nodes, opening up the field of WSNs [2]. To allow infrastructure-less ad hoc deployments, sensor motes have to be self-powered, which shifts energy efficient operation into the focus when reaching for long network lifetimes [6]. Given the fact that wireless communication typically accounts for a large fractions of the overall power consumption [15], the transceiver and especially the MAC layer were the subject of many studies [16].

### 2.1 Energy-Efficient MAC

The key concept for low-power MAC protocols is *duty-cycling*. With duty-cycling, the transceiver is switched on and off following a schedule defined by the MAC protocol. This mechanism saves energy from idle-listening, which would otherwise consume considerable amounts of energy [15]. Duty-cycling protocols can be divided into synchronized [8] and unsynchronized [17], [18] approaches. With synchronized algorithms, the nodes align their duty cycles to agree on possible time slots for transmissions. Such approaches make sending less costly, but introduce signaling overhead to establish a common time base. Using unsynchronized protocols, a sending node does not know about the duty cycle of its neighbors and, therefore, has to extend its transmissions to assert that each node woke up and got a chance to receive the frame, making transmissions costlier in terms of both energy and channel utilization.

In that context, a note on terminology is very important: When discussing MAC protocols, many authors refer to a *preamble* as a signal that is used to wake up duty-cycled nodes, announcing a transmission. Such mechanism is, for example, required with unsynchronized MAC protocols. This use of a *preamble* has to be clearly distinguished from the physical layer preamble, sometimes also called physical layer training sequence, which we discuss in this paper. A receiver uses the physical layer preamble to synchronize on the signal by estimating parameters like frequency and clock offsets. Our approach, which allows saving the overhead of preamble symbols, is a pure physical layer concept and, therefore, independent from MAC layer algorithms. In fact, our idea can be complemented with duty-cycled MAC protocols.

Another strategy to save energy is to use transmit power control [7]. Intuitively, a transmitter can use lower transmit power when addressing nodes with low channel attenuation, such as nodes in close proximity or with unobstructed line of sight. By decreasing the transmit power, the sender can save energy, which would otherwise be wasted in the power amplifiers. A potential drawback is that the sender needs feedback to estimate the channel attenuation, making it a good candidate in relatively static scenarios with bidirectional communication.

A more recent trend is to drop support for complex mesh network topologies, which are, for example, part of IEEE 802.15.4 and stick to simpler star networks. This is especially visible with the very successful Bluetooth Low Energy (BLE) standard, whose energy efficiency results mainly from changes on the link and network layer [19]. Even though the physical layer was slightly adapted towards larger channel bandwidths, the main improvements stem from simplified network structures that allow saving energy through less overhead in maintaining connections.

### 2.2 Ultra Low-Power Communication

More recent advancements, sometimes called *ultra low-power communications*, allow for even smaller sensor motes that support network lifetimes of up to several years. Such motes are based on wake-up receivers that are most of the time completely switched off (or, to increase sensitivity, in a very low-power mode) and only activated from actual transmissions [20]. The basic principle is similar to Radio-Frequency Identification (RFID) [21], where the transmitted signal induces current in the receiver, which is used to wake up the communication module.

The possibility to activate the radio module only during actual transmissions could be regarded as the ideal duty-

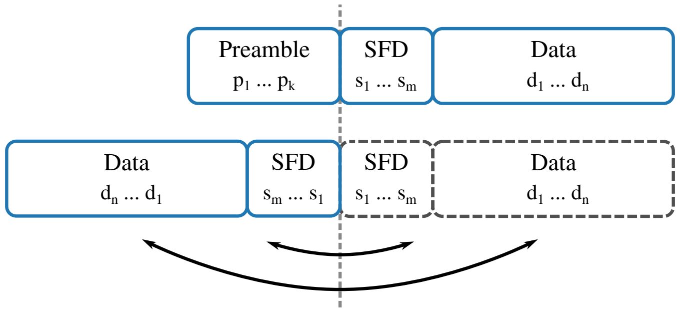


Figure 1. Comparison of normal frames (top) and mSync (bottom). Instead of a preamble, we use the data to synchronize and calculate local estimates for parameters like frequency offset, sampling clock offset, and symbol timing. This allows omitting the preamble and reduce the frame size, as only the components with the solid outline are sent over the air.

cycle and allows us to overcome idle listening all together. This technology is an enormous step that opens the field for completely new applications. In wildlife monitoring, for example, we can use much smaller batteries and build light-weight sensor motes that can even be mounted on small, flying animals like bats [4] or crows [5].

Another idea, also adopted from RFID, is the use of backscattered signals for communication [22], [23]. Here, tags can be completely passive as they do not have to generate a signal locally, but merely reflect the signal of an interrogator. The actual information is encoded by changing the antenna's impedance, which varies the amplitude of the reflected signal. While this is an interesting concept, it cannot be directly applied to WSNs, as it relies on a very capable interrogator.

However, with *ambient backscatter* [22] and bistatic scatter radio [23], this concept was extended and applied to decentralized low-power networks such as WSNs. Instead of using an interrogator signal, these works use ubiquitous signals from broadcast radio, TV stations, or dedicated carrier emitters. Such tags are not completely passive, but the power consumption is greatly reduced since the signal does not have to be generated locally, saving the energy that is otherwise used in power amplifiers. Very recently, a similar concept was applied to IEEE 802.11b allowing for, what the authors call, *Passive WiFi* [24]. By modulating a WiFi frame from another device, they are able to produce standard compliant IEEE 802.11b frames, while reducing the power consumption by orders of magnitude.

To summarize, we presented a broad spectrum of ideas and concepts to design energy efficient wireless networks. With our approach, we introduce another, orthogonal option that can be beneficial for many low-power systems. By removing the preamble without degrading physical layer performance, mSync can complement the presented approaches and offers a new strategy to further optimize energy consumption of low-power transceivers.

## 3 Preamble-less Frame Detection and Decoding

Before describing mSync, we should first recap the typical physical layer frame format. As illustrated at the top of Figure 1, it consists of a preamble, followed by the Start of Frame Delimiter (SFD) and the actual data payload. In the receiver, the preamble is used to synchronize on the frame by deriving local estimates for parameters like frequency offset, sampling clock offset, and symbol timing. This is typically done with a feedback loop that locks on the signal by adjusting the parameters according to an error signal [25]. With symbol timing recovery, for example, the sampling points are gradually adjusted towards the largest opening in an eye diagram. This stage usually employs blind estimators that work without prior knowledge of the data. For our algorithm, it is irrelevant which exact algorithm or combination of algorithms is used in the receiver. The important aspect is that they adjust during reception, which is usually the case, also in state of the art algorithms [26].

When designing a new physical layer, it can be challenging to find a parameter set that offers a good trade-off between performance and overhead in terms of preamble symbols. Ideally the receiver would synchronize exactly at the end of the preamble and reliably detect the SFD. In practice, it involves a trade-off between the overhead (i.e., length of the preamble and, thus, time to lock) and the reliability of frame detection. Using a longer preamble, the receiver has more time to synchronize and chances are higher that the SFD can be detected.

### 3.1 Reversed Frame Structure

While balancing this trade-off for a custom highly-optimized receiver, we thought of using a different frame format that offers important advantages: it relaxes the requirements for the synchronization algorithm, while, at the same time, reduces the frame size, improving the energy consumption. The core idea is to omit the preamble and adapt the frame format shown at the bottom of Figure 1. In contrast to normal frames, we send only the data symbols followed by the SFD, which might, in this case, better be called an end of frame delimiter. We will, however, stick to the term to emphasize its correspondence with the SFD in a normal frame. When looking at Figure 1, it is important to note that the dashed boxes are not sent over the air but added to illustrate the concept. Only the solid boxes depict the parts of the signal that are actually transmitted. From the figure, we can see that mSync allows us to reduce the frame size, while carrying the same amount of data.

The most interesting part of mSync is the decoding strategy for this modified physical layer frame format. Given the fact that the receiver is not synchronized when the first data symbols arrive, it cannot decode the data directly. Instead, it locks on the frame by processing the signal using the same blind estimation algorithm as the normal receiver. The process might, however, require more input, as it uses data symbols instead of a preamble sequence that is optimized for fast convergence of the synchronization algorithm.

The crux of mSync is that it buffers samples, for example, in a ring buffer. This allows us to process the signal twice; once to lock on the frame and another time to decode it. Using a simple implementation, the buffer size has to be set according to the number of samples corresponding to a maximum sized frame plus the SFD. We will later discuss ideas on how to reduce the amount of data that has to be buffered.

Using the data to synchronize, the algorithm has much more time, i.e., the whole duration of the data symbols, to lock on the frame. When the receiver is synchronized, it can recognize the SFD at the end of the frame. If that happens, the receiver keeps its state, i.e., its current estimates of the signal parameters and, instead of continuing with the normal sample stream, it processes the ring buffer in reverse direction and decodes the data. During this reverse operation, the receiver traverses the sample stream again, processing the samples in the order indicated by the dashed boxes in Figure 1. This process can also be thought of *mirroring* the received signal in time domain at the dashed vertical line. The name mSync, for mirror synchronization, is derived from this central characteristic of the algorithm.

Decoding the signal while traversing the buffer backwards also explains why we send the over-the-air signal reversed (i.e., why we change the samples of the data from $d_1, \cdots d_n$ to $d_n, \cdots d_1$). This is not strictly necessary, but it asserts that the output of mSync (the boxes with the dashed outline) correspond to the output of a normal receiver (cf. the right hand side of Figure 1). This has the advantage that the decoder outputs the exact same bit sequence as a normal receiver, which eases integration of our algorithm. Extending a receiver with mSync is, therefore, straightforward and merely comprises replacing the synchronization algorithm. The other components can be left unchanged.

The part of the algorithm that we did not discuss yet is how the receiver stays locked while switching directions. Since this depends on the actual algorithm that is used in the receiver, we can only discuss exemplary implementations. In the following, we describe how our algorithm can be used with the Mueller and Müller (M&M) algorithm [27] for timing recovery. While the M&M algorithm is not state of the art, it serves as a good example to describe the concept. However, we argue that the general idea can also be applied to more complex state of the art algorithms like symbol timing recovery with polyphase filter banks [26]. The M&M algorithm implements a feedback system that performs timing recovery and estimates the sampling clock offset, i.e., it calculates estimates for the number of samples per symbol $\Omega$ and the position of the $n$-th symbol in the sample stream $B[n]$. If we consider a real-valued binary signal that encodes its data with $\{1, -1\}$, the algorithm calculates the error feedback $e$ after decoding the $n$-th symbol as

$$e[n] = \hat{B}[n-1]\, B[n] - \hat{B}[n]\, B[n-1], \tag{1}$$

where $\hat{B}$ is the, probably corrupt, decoded symbol with $\hat{B} \in \{1, -1\}$. This error signal is used to adjust the estimates $\Omega$ and $B$ as

$$\Omega[n+1] = \Omega[n] + g_\Omega\, e[n] \tag{2}$$
$$B[n+1] = B[n] + \Omega[n] + g_B\, e[n]. \tag{3}$$

Here $g_\Omega$ and $g_B$ are gains for the error feedback of the corresponding parameter that can be used to adjust the sensitivity of the controller.

An exemplary iteration of the algorithm is illustrated in Figure 2, which shows the noise free analog signal as a dotted line. The actual symbol timing, unknown to the receiver, is indicated by the solid vertical lines. The sampling points of the SDR are marked as crosses, while the dots indicate the points that the receiver considers for decoding (i.e., the estimated symbol timing). Since the estimated symbols are not exactly at sample positions, the receiver interpolates the values with a minimum mean squared error FIR interpolator. When using mSync, the receiver uses the very same algorithm, but, in addition, stores the samples in a ring buffer and compares the decoded values $\hat{B}[n-m+1], \cdots, \hat{B}[n]$ with the reversed SFD $s_m, \cdots, s_1$ (cf. Figure 1). Once they match, the receiver continues processing samples from the buffer. To foster compatibility with the normal receiver, we regenerate the preamble bits in the receiver and prefix them before every frame, just as if it would have been received over the air. This way, mSync is completely transparent for the rest of the receiver.

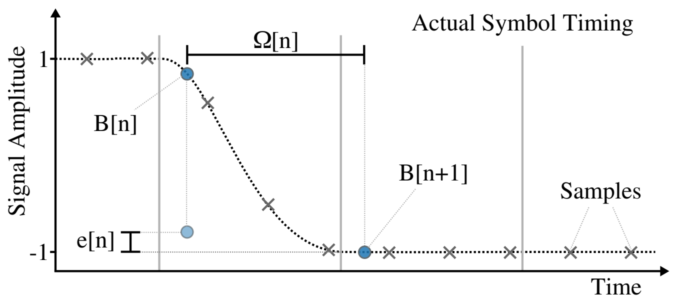


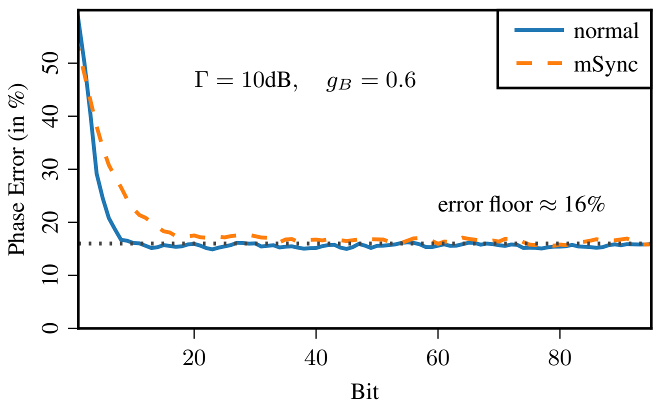
Figure 2. Illustration of the Mueller and Müller algorithm for timing recovery and clock offset estimation.

## 3.2 Synchronizing on Data

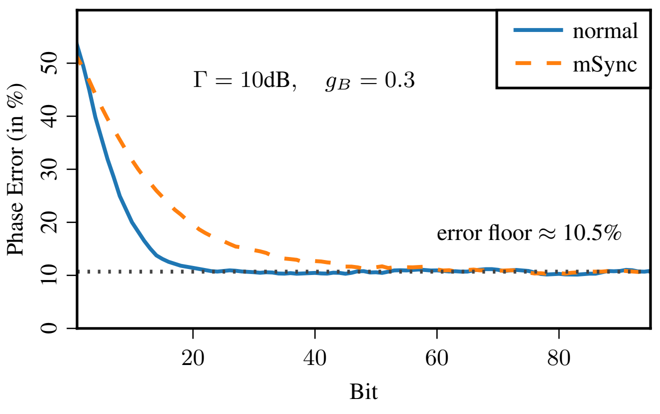
One important difference of mSync is the use of an unknown data signal instead of optimized preamble sequences for synchronization. For frame-based systems that need the receiver to re-synchronize on every frame, this could, in theory, cause problems. Depending on the physical layer and the receive algorithm, not all bit sequences might be equally suited to derive signal parameters. A Binary Phase Shift Keying (BPSK) signal that is all ones or all minus ones, for examples, cannot be used to extract symbol timing information. Fortunately, this is unlikely to happen, since a state-of-the-art physical layer uses a scrambler if the data tends to include long strings of ones or zeros. Also from an information theoretic perspective, the physical layer will be tuned towards equally probable symbols to maximize entropy and, thus, self-information.

To better understand the implications of using mSync, we study how locking on data symbols differentiates from ideal preamble sequences. Our motivation for this study is twofold. First, we want to understand how much longer it takes to synchronize on a random data pattern. Second, we want to highlight another potential benefit of mSync. Using the data to synchronize, we have much more time to lock on the frame. This allows us to reduce the gain of the error feedback in the controller, which slows convergence of the synchronization algorithm, but also reduces the noise feedback of the controller, making it more stable.

To show this effect, we set up simulations to compare the ideal locking sequence for the M&M algorithm (i.e., $00110011\cdots$) with a random bit pattern. Similar to our



(a) Faster convergence.



(b) Slower convergence.

Figure 3. Impact of the error feedback gain on the time that it takes to lock. With mSync, we have more time to synchronize, allowing us to choose a setting with slower convergence, but lower error floor.

BATS transceiver (described in more detail in Section 5), we produce a BPSK signal with five samples per bit and apply a matched filter. The resulting sample stream is passed through an Additive White Gaussian Noise (AWGN) channel to produce a signal with a given target Signal to Noise Ratio (SNR) $\Gamma$. To rule out interactions from level controllers and to isolate the effect of locking on the data, we scale signal and noise, such that the average signal $S$ plus the average noise power $N$ equals unity

$$S = \frac{\Gamma}{1+\Gamma}, \quad N = \frac{1}{1+\Gamma}. \tag{4}$$

To keep the example simple, we fix $\Omega$ to the correct value and record the average phase error dependent on the sample of the frame. The average phase error after the n-th bit for 100k frame transmissions at an SNR of 10 dB are depicted in Figure 3. With $\Omega = 5$ the maximum phase error is 2.5, which we set as 100 %. In Figure 3a, we show a configuration with a rather high error feedback ($g_B = 0.6$) and fast convergence. Using a normal frame with an optimized preamble sequence, the phase error stabilizes fast (after only about 8 bit). Such configuration might be used in a typical physical layer, where the preamble should be as short as possible to reduce overhead. As expected, mSync needs more time to lock on a frame. In this setup, it reaches the error floor after about 20 bit.

In Figure 3b, we show results for the same configuration, but with reduced error feedback ($g_B = 0.3$). While locking is slower in that configuration, the general trend is similar. mSync needs more time to lock than an optimized preamble (about 25 bit for a normal frame compared to about 50 bit for mSync). The advantage of the slower configuration is its higher stability through lower noise feedback. In the slower configuration, the phase error reaches an error floor of only 10.5 % compared to 16 % with the faster configuration. Such configuration is clearly beneficial, but might not be suitable in a normal receiver. The longer convergence time, would ask for a longer preamble and, therefore, increase the overhead per frame, a problem that we do not face with mSync.

To summarize, the results highlight mSync's potential to use more stable controller configurations with lower noise feedback. While the quantitative results of these experiments are valid for the M&M algorithm, we expect similar qualitative behavior also for other synchronization algorithms.

## 4 IMPLEMENTATION

To study our algorithm and to show the feasibility of the approach, especially that it is possible to stay locked while switching directions, we implemented the algorithm for GNU Radio, a real-time signal processing framework for use in SDR systems [28]. In contrast to, for example WARP [29], GNU Radio implements signal processing on a General Purpose Processor (GPP), like a normal PC, which lends itself well for rapid prototyping [30]. Using GNU Radio, signal processing is split in *blocks* that implement specific functions like filters, resamplers, and modulators. To exploit modern multi-core CPUs, signal processing is parallelized by starting each block in a separate thread. Compared to iterative programming environments, like MATLAB, parallelized processing adds complexity, but it is the central design point that enables real-time operation.

In GNU Radio, a transceiver is realized with a *flow graph*. It defines a specific configuration with a set of blocks, their parameters, and their connections. To get a better idea of the concept, Figure 4 shows a screenshot of the relevant parts of our BATS receiver in GNU Radio Companion, a graphical frontend to setup and configure GNU Radio flow graphs. Integrating our algorithm in the existing receiver was straightforward. We merely had to change the blocks in the shaded area, which contains the logic to switch between the legacy M&M implementation and mSync. With the *Selector* blocks, we can pipe the incoming sample stream either through the normal (top) or the mSync implementation (bottom). The other blocks of the receiver can be left unchanged. They are used to demodulate the differential BPSK signal to a binary stream, search for the preamble sequence, and, once found, process the data payload.

GNU Radio already comes with two very similar implementations of the M&M algorithm; one for complex and one for real signals. We implemented our algorithm for both versions, since the BATS receiver uses the complex variant, while the IEEE 802.15.4 receiver uses the real one.

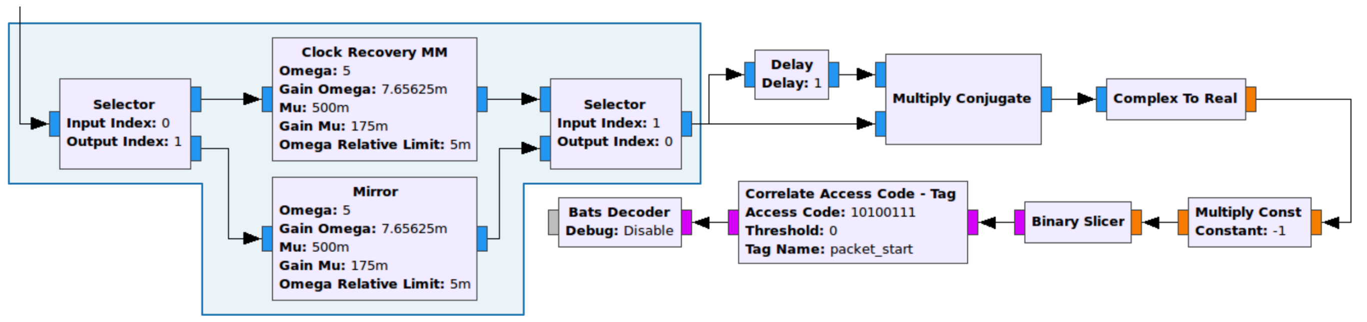


Figure 4. The relevant part of our ultra low-power receiver in GNU Radio Companion. To support both the normal preamble as well as our preamble-less reversed frame format, we merely had to introduce the possibility to switch the clock recovery algorithm (shaded area).

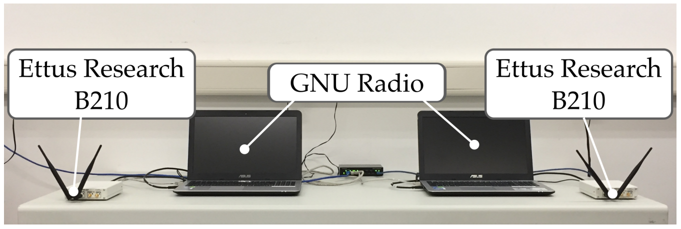


Figure 5. The experiments are conducted in an office environment, using GNU Radio together with Ettus Research B210 SDRs.

Apart from rapid prototyping, an important advantage of GPP-based SDR architecture like GNU Radio is the possibility to use the same code for simulations as well as experiments. Since signal processing is implemented on the PC, we can either loop back the samples from the transmitter into the receiver to perform simulations or connect SDRs for over-the-air experiments. In this paper, we use this feature and evaluate both transceivers by means of simulations over an AWGN channel and real transmissions in an office environment.

## 5 Case Study 1: Bats Transceiver

The first use-case of our algorithm is the Bats transceiver, a custom ultra low-power transceiver, that we developed to track mouse-eared bats (*Myotis myotis*) in their natural habitat [4].[1] With a weight of about 20 g, these bats can only be equipped with sensor motes of up to 2 g. In our project, we used this weight budget for a 1 g mote that is powered by a 1 g battery. This weight constraint has distinct impact on wireless communications and is asking for a custom application-specific design. The main limitation of out platform is that the battery does not supply enough current to drive the wireless transceiver directly. Instead, we have to use the battery to charge a capacitor, which is then used to power the transceiver and send short bursts of data. More details and rationale on the design and implementation of this ultra low-power transceiver are available in prior work [4], [31].

In the context of this paper, the important aspect is that the maximum frame size is limited by the energy stored in the capacitor. Furthermore, we are restricted to simple modulation and coding schemes to keep the sensor mote complexity at a minimum. To assess the performance of this application-specific physical layer, we developed SDR-based prototypes that we used for simulations and experiments in realistic environments [32].

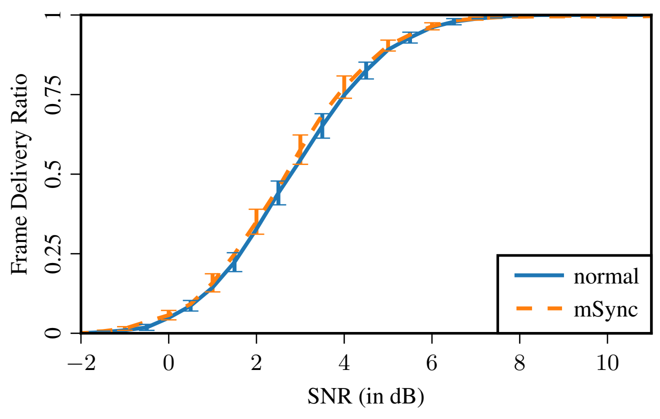
1. http://www.for-bats.de/

### 5.1 Physical Layer Performance

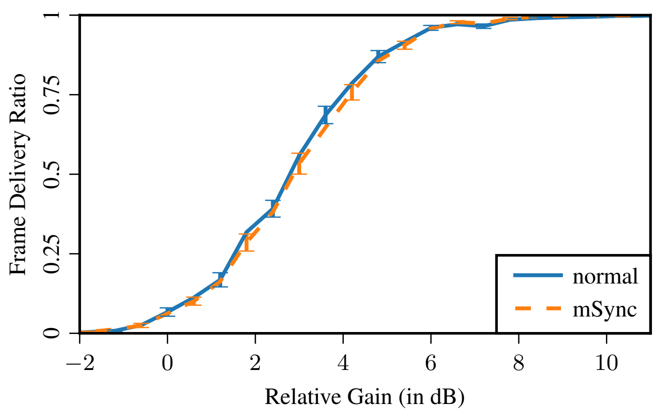
Using differential BPSK, the energy in the capacitor is just enough to send 12 byte frames at a data rate of 200 kbit/s. Without mSync, a frame comprises a 2 byte preamble, a 1 byte SFD, and a 2 byte Cyclic Redundancy Check (CRC), leaving 7 byte for the payload. Given the short frames size, we stick to error detection using the CRC, but do not employ forward error correction. The SDR implementation uses $\Omega = 5$ samples per symbol, which corresponds to a sample rate of 1 Msps. The other parameters of the M&M algorithm were set to the default values used in GNU Radio version 3.7. Finally, we use a moving average over the duration of five bits to normalize the signal to an average power of one, before feeding it to the clock recovery algorithm.

When conducting simulations using the GNU Radio implementation of our new algorithm, the first observation is that it works in the first place. This serves as a proof-of-concept for mSync, but, of course, we wanted to go further and provide a quantitative analysis of the impact on physical layer performance. For this, we conduct simulations where we send frames with a pseudo-random payload over an AWGN channel and measure the frame error rate. In corresponding runs, i.e., with and without our algorithm, we use the same channel coefficients and payloads. Furthermore, we set the noise to an average power of one and adapt the amplitude of the signal to reach the desired SNR. The resulting frame delivery ratio at different SNR levels is depicted in Figure 6a. The error bars in this and the following plots depict the 95 % confidence intervals. For better readability, we alter between plotting the confidence intervals for the configuration with normal frames and the configuration with mSync.

The results show that we start receiving frames at around 0 dB and reach 100 % at about 7 dB. More interestingly, the graph shows that the two modes behave exactly the same, which is a very positive result. It shows that the improved energy consumption of mSync (through shorter frames without preamble symbols) does not have to be traded for physical layer performance. This is the best possible outcome, since the goal was not to improve the frame error



(a) Simulations over an AWGN channel.



(b) Experiments in an office environment.

Figure 6. Observed packet delivery ratio of the BATS PHY in simulations and measurements.

rate, but to benefit from shorter frames without *degrading* physical layer performance.

To rule out potential simplifying assumptions in our simulations, we set up real over-the-air measurements. With GNU Radio, switching between simulations and real experiments is straightforward. The possibility to use the same code in simulations and measurements is a big advantage, as it allows us to directly compare the results. Figure 5 shows our measurement setup. In this and the following experiments, we do not employ any unrealistic simplifications (like using a common clock source for sender and receiver). The oscillators in sender and receiver run completely independently, leading to typical hardware impairments like frequency offsets and sampling clock offsets. We use two B210 SDRs from Ettus Research, which we configured to send in the 868 MHz band. While we cannot strictly rule out interference, we chose a frequency that seemed to be vacant in our lab. To set different SNR levels, we kept the receive gain constant and varied the output power of the transmitter. Since the B210 is no calibrated measurement device, this method allows us to change the relative SNR, but not to configure an absolute level. Like in the simulations, we use 5 samples per symbol, resulting in a sample rate of 1 Msps.

The results of the experiment are depicted in Figure 6b, where we plot the frame deliver ratio at different SNRs. For better comparison, we shifted the relative SNR on the x-axis to values corresponding to our simulations. The graph shows that the experiments are in perfect compliance with our simulations. Again, mSync and normal operation do not show any differences, highlighting that our approach does not degrade the frame error rate. The graph shows two slight dips in the curve, which can be explain by nonlinearities of the amplifiers. This is in accordance with the datasheet of the Analog Devices AD9361 transceiver used with the B210.

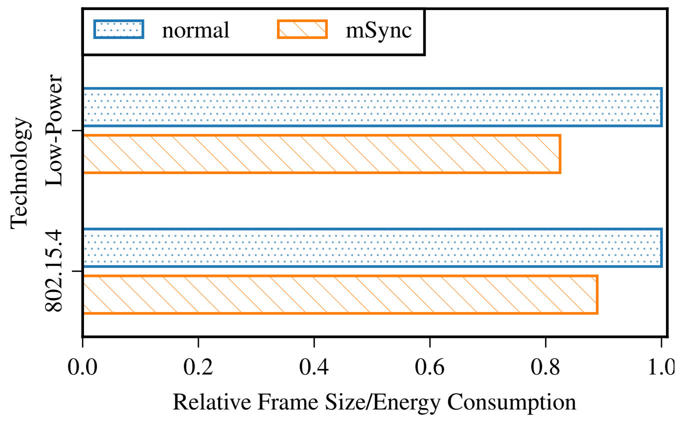


Figure 7. Comparison of the relative frame size/the relative energy consumption between normal frames and mSync. For IEEE 802.15.4, the graph shows 30 byte packets, as used in our simulations and measurements.

## 5.2 Energy Savings

For our custom BATS transceiver, mSync allows us to shorten the frame by 17 % (through saving the 2 byte preamble of the 12 byte frame). This improvement is visualized at the top of Figure 7, where we plot the frame size of mSync relative to a normal frame. In our particular case, the shorter frames translate directly into energy savings in the *transmitter*, as the sensor mote's main task is to send periodic beacons to the ground network. Apart from this specific case, also other communication modules could benefit from mSync, especially when the analog radio front end (in particular the power amplifiers) are responsible for a large part of the energy consumption. For the *receiver*, the energy consumption is not straightforward to quantify, since the algorithm introduces a slight computational overhead and requires to buffer samples corresponding to the maximum frame size. A detailed comparative study between an SDR implementation of a normal receiver and mSync is, therefore, presented separately in Section 8.

While optimized energy consumption per frame is already a good argument for our algorithm, wildlife monitoring is a prime application also for other reasons. WSNs for wildlife monitoring are usually heterogeneous with energy-constraint mobile nodes on the animals that send data to more capable stationary nodes. These stationary nodes are typically SDR-based, implementing a custom and application-specific physical layer. In such scenarios, our algorithm is straightforward to apply: SDRs are easy to extend with the required functionality and the mobile nodes rely on a

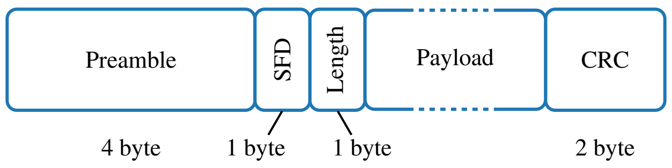


Figure 8. Structure of an IEEE 802.15.4 frame. The payload size can be up to 125 byte.

custom design either way. With these modifications, the more capable stationary node can use the slightly more complex algorithm, while the mobile node can benefit from considerable improvements in terms of energy consumption.

## 6 Case Study 2: IEEE 802.15.4 Transceiver

Motivated by the promising results of the Bats transceiver, we were curious to apply the concept also to more complex transceivers. We chose the IEEE 802.15.4 O-QPSK physical layer for the 2.4 GHz band, which forms the base of the ZigBee stack. This physical layer is designed to provide energy efficient communication for WSNs and IoT applications and could, therefore, greatly benefit from the energy savings provided by our approach.

### 6.1 Physical Layer Performance

Fortunately, there is already an Open Source implementation of IEEE 802.15.4 available for GNU Radio. This implementation was started by Thomas Schmid [33] and later overhauled by us in [34].[2] Based on O-QPSK, the IEEE 802.15.4 physical layer is slightly more complex. To encode the data, the transmitter maps each group of 4 bit to one of 16 pseudo-noise chip sequences. These 32 bit chip sequences are then O-QPSK modulated to create a signal with a chip rate of 2 Mcps. In the SDR transceiver, we process the signal with a sample rate of 4 Msps.

The IEEE 802.15.4 GNU Radio module already uses the M&M algorithm. Integration of our algorithm into the receiver is, therefore, straightforward. As shown in Figure 4, we merely have to replace the M&M block with our modified version. All parameters and the other components of the receiver are left unchanged. The frame format of a normal IEEE 802.15.4 frame is shown in Figure 8. Each frame consists of a 4 byte preamble, 4 byte physical layer overhead (for SFD, a header, and the CRC), and the data payload of up to 125 byte. Using our algorithm allows us to save 4 byte of all frames, independent from the their total size. For an ACK with a total length of 11 byte, this corresponds to 36 %. But even for a full-sized frame with a total length of 133 byte, the improvement is still 3 %.

Similar to the previous use-case, we start our evaluations with simulations over an AWGN channel. We send 30 byte frames with a pseudo random payload and record the frame delivery ratio. The relative improvement of mSync for a 30 byte frame is depicted in Figure 7. With mSync, we can reduce the air time by 11 % from 1.15 ms to 1.02 ms, while transmitting the same data. The results of the simulations are depicted in Figure 9a, where we plot the frame delivery ratio for different SNRs. As in previous figures, the error bars indicate the 95 % confidence intervals. Like with the Bats transceiver, the main observation is that the modes behave exactly the same. That means that also with this transceiver, we can benefit from mSync without degrading physical layer performance.

Again, we wanted to back up our results with real experiments and conducted measurements in an office environment. We used the same B210 SDRs, this time transmitting in the 2.4 GHz band. Since this band is very crowded, we chose a channel at the upper end, as there are no WiFi networks allowed in our region. Using this part of the spectrum, we were able to avoid most interference sources and had stable experimental conditions. Given the previously discussed limitations of the B210, we plot the relative SNR and align the x-axis for better comparability with the simulation results. The resulting graph is shown in Figure 9b. Also in this experiment, the results match very well with simulations, proving that mSync is feasible in practice and that we can benefit without any drawbacks in terms of physical layer performance.

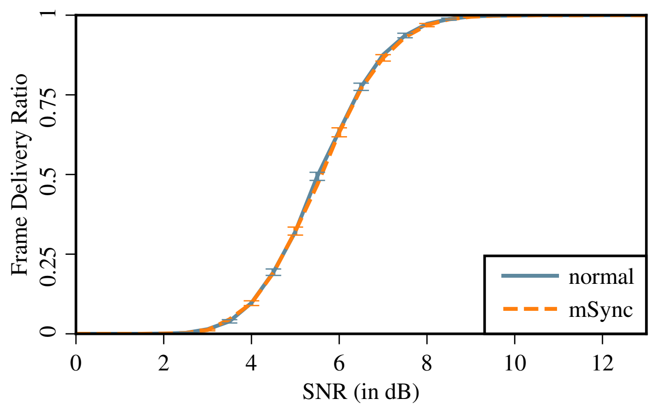
### 6.2 Impact on MAC Layer Goodput

While optimizing the energy consumption was the main motivation to introduce mSync, shorter frames also reduce occupancy of the wireless channel, potentially improving network goodput. For our Bats transceiver, this aspect is not of prime interest, as infrequent transmissions lead to low network utilization. For IEEE 802.15.4, in contrast, the maximum achievable goodput can be a relevant aspect. Quantifying the impact of shorter frames is, however, not straightforward, as the relationship between the overhead per frame and network goodput is non-trivial. The complexity stems from the slotted operation of the channel access algorithm and the fact that we use the channel more efficiently once we access it.

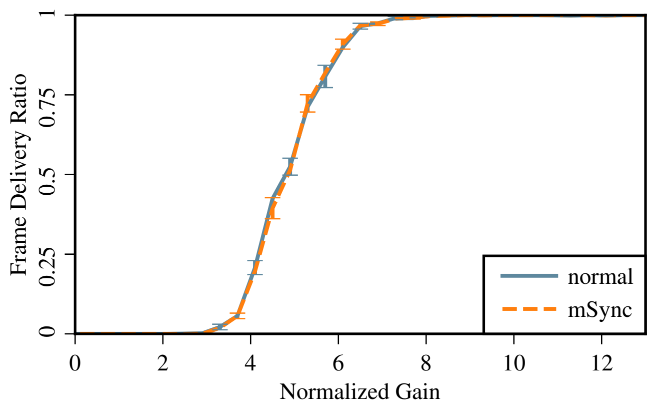
To study the possible improvements of our algorithm, we employ the Markov model presented in [35]. This model considers the stationary throughput of a saturated IEEE 802.15.4 network. While the standard defines several network topologies and modes of operation, we focus on a typical network, consisting of a Personal Area Network (PAN) coordinator that orchestrates nodes in a star topology. In such networks, the coordinator establishes a superframe cycle that is used to subdivide time into a Contention-Free Period (CFP) and a Contention Access Period (CAP). The channel access during the CAP uses slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with a slot length of *aUnitBackoffPeriod* that has a duration of 20 symbols. For the sake of brevity, we only consider the CAP and unacknowledged transmissions in our scenario. The model is, however, straightforward to extend to acknowledged transmissions and unsaturated conditions, as shown in [35]. Furthermore, the battery life extension mode, which allows reducing the size of the initial backoff window, is considered to be disabled.

Similar to Wireless LAN (WLAN), IEEE 802.15.4 uses an exponentially increasing backoff algorithm for channel access. The backoff windows starts at $2^{macMinBE}$ slots, which are doubled in each round until it reaches a maximum size

2. https://www.wime-project.net/



(a) Simulations over an AWGN channel.



(b) Experiments in an office environment.

Figure 9. Observed packet delivery ratio of the IEEE 802.15.4 PHY in simulations and measurements.

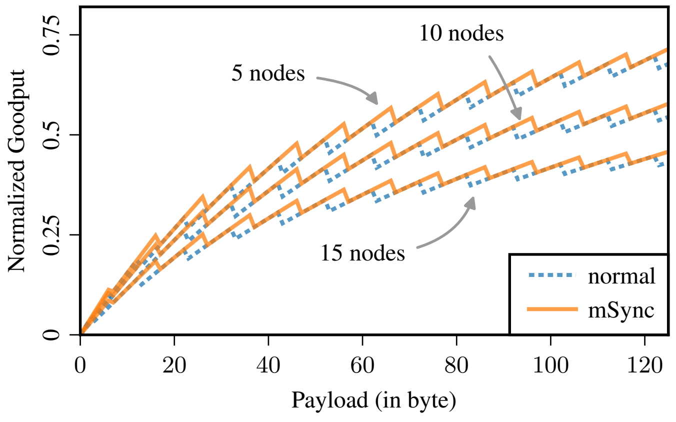


Figure 10. Normalized saturation goodput for our approach and normal frames for typical network sizes and an increasing payload length.

of $2^{macMaxBE}$ slots. Frame transmission fails, if the MAC went through *macMaxCSMABackoff* rounds without finding the channel idle when trying to send. In that case, the frame is dropped, while the algorithm restarts with the next frame in the queue. The most important MAC layer parameters are also summarized in Table 1. All parameters correspond to their default values given in the standard [13].

The Markov model that we use to calculate the saturation goodput is much inspired by Bianchi's seminal work on the Distributed Coordination Function (DCF) of IEEE 802.11 [36]. Compared to WLAN, the main difference is that sensor nodes enter a power saving state during their backoff periods and, therefore, do not recognize if the channel turns busy while they are waiting. After the backoff, nodes sense the channel twice in two consecutive slots and transmit only if the channel is sensed idle in both slots. To incorporate this mechanism in the Markov model, Bianchi's key assumption of constant and independent collision probabilities had to be adapted to a constant and independent probability $\phi$ that a node starts sensing the channel in a randomly selected slot. At that first channel assessment, the node senses the channel busy with a probability $\alpha$. If the channel is free, the node continues with the second channel assessment during which it senses the channel busy with a probability $\beta$.

The detailed derivation of $\alpha$, $\beta$, and $\phi$ is available in [35]. In our context, the most interesting result from the paper, is the normalized saturation throughput $S$ that defines the fraction of time slots spent to transmit non-colliding frames that contribute to the throughput of the network. It is calculated as

$$S = LN\phi(1-\phi)^{N-1}(1-\alpha)(1-\beta)\,, \tag{5}$$

where $N$ represents the number of nodes in the network and $L$ the length of a frame in time slots. Rounding the size of a frame to a complete backoff slot, we use the payload size $P$ and the overhead $O$ of a frame to calculate its length as

$$L = \left\lceil \frac{(P+O)R_S}{aUnitBackoffPeriod} \right\rceil\,. \tag{6}$$

Both $P$ and $O$ are given in byte and are converted to symbols with a rate of $R_S = 2$ symbols per byte. As payload, we consider the data in the MAC layer, denoted as *Payload* in Figure 8, i.e., the MAC frame excluding the CRC. At this point, it is important to differentiate between throughput and goodput to understand the advantage of our approach. By omitting the preamble, we reduce the overhead from 8 byte per frame (4 byte preamble, 1 byte SFD, 1 byte length field, 2 byte CRC), to 4 byte. If this difference allows us to save a slot, we use the channel more efficiently as the average goodput per slot $G$ is increased. We calculate $G$ as

$$G = \frac{PR_s}{aUnitBackoffPeriod} \cdot \frac{1}{L}\,. \tag{7}$$

Scaling the normalized throughput $S$ with the efficiency of the channel access $G$, we calculate the normalized goodput for all payload sizes of up to 125 byte. The results for networks of 5, 10, and 15 nodes are depicted in Figure 10. In

Table 1
Most relevant MAC layer parameters.

| Parameter | Value |
|---|---|
| *aUnitBackoffPeriod* | 20 Symbols |
| *macMinBE* | 3 |
| *macMaxBE* | 5 |
| *macMaxCSMABackoff* | 4 |

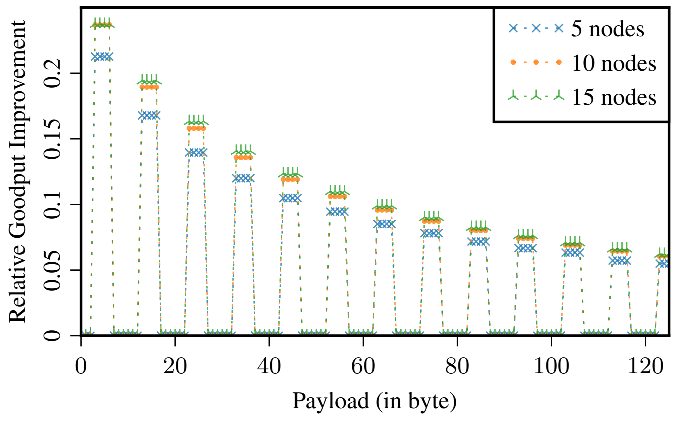


Figure 11. Relative goodput improvement of our approach over normal frames for typical network sizes and an increasing payload length.

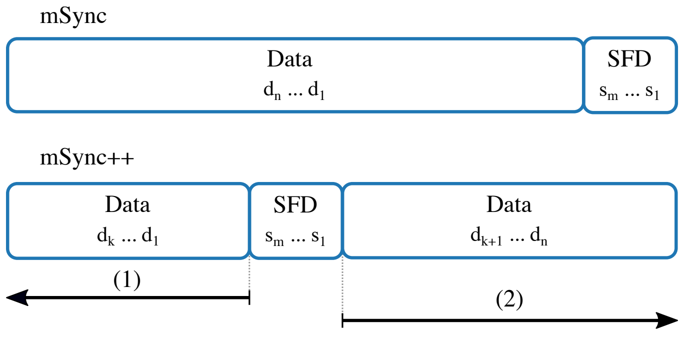


Figure 12. Comparison of a normal mSync frame (top) and an optimized version that requires less buffering in the receiver (bottom). With the optimized version, the SFD is inserted after a fixed number of byte $k$. The resulting frame is first decoded in backwards direction (1) and then like a normal frame in forward direction (2).

accordance with [35], we can see that the goodput decreases with the number of nodes as well as with smaller frame sizes. Most importantly, however, we can see that our approach performs better or at least equally good, with a rather constant difference to normal frames.

The regular zigzag pattern stems from the slotted operation of the MAC layer, which uses slots with a duration corresponding to 10 byte. Since a frame blocks the channel always for full slots, and since we save 4 byte from the preamble, we benefit only in 4 of 10 cases.

To quantify the gain, we provide another view on the data in Figure 11, where we plot the relative improvement over normal IEEE 802.15.4 frames. The graph shows that, especially for smaller payload sizes, the gain is significant with increases of over 20 %. Even for large frames an improvement of about 5 % can be achieved. Furthermore, we can see a higher improvement for larger networks, which, however, stems from their lower network goodput as opposed to higher absolute gains.

To summarize, these analytical evaluations highlight that our approach, which was initially motivated by optimizing the energy consumption of sensor motes, can also provide considerable improvements in terms of network goodput.

## 7 Minimizing Signal Buffering

We are aware that the need to buffer samples in the receiver is the main drawback of our approach. With the presented version of mSync, the amount of data that has to be buffered would grow linearly with the signal bandwidth and the maximum frame size. This could easily render our approach unfeasible in practice, especially when implementing it on an integrated platform.

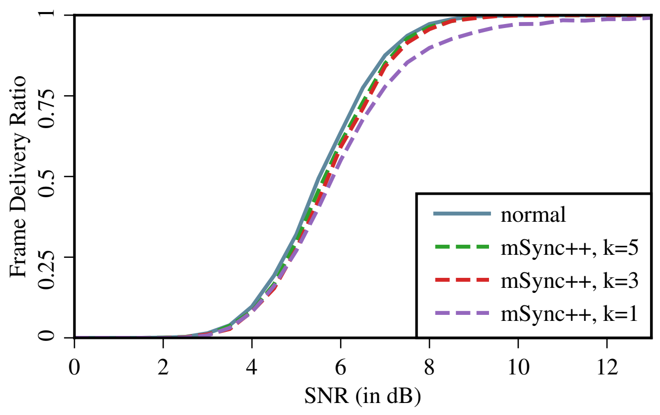
To address this issue, we introduce mSync++, a straightforward extension of our algorithm. It is motivated by the observation that the main problem with long frames is that the receiver has to buffer the whole frame, even though it is likely synchronized already after a few data symbols. In Section 3.2, we have seen that even with more stable and slower converging configurations, the synchronization algorithm locks after a few byte. Therefore, with mSync++, we do not put the SFD at the very end of the frame, but place it latest after a fixed number of byte $k$. The resulting frame format is depicted in Figure 12. To decode the frame, we start like with normal mSync. We use the data to synchronize and search for the SFD. Once found, we store the state of the receiver, and traverse the signal in reverse direction, while decoding the data. The difference is that we only have to go backwards for $k$ byte. After that, we restore the state of the receiver to the state when we detected the SFD and continue in forward direction. The frame is designed so that the decoded bits match with a normal frame, allowing us to leave the rest of the receiver unchanged.

Since $k$ is a fixed system parameter, the number of samples that are stored locally do not depend on the maximum frame size, allowing the algorithm to scale. To optimize the receiver, $k$ could be used to balance the trade-off between the amount of data that has to be buffered and the probability that the receiver is synchronized at the SFD. In other words, it balances the performance of the receiver against its complexity. With lower $k$, the SFD is placed earlier in the frame, decreasing the data that has to be buffered. However, lowering $k$ also gives the receiver less time to lock the signal, potentially degrading performance.

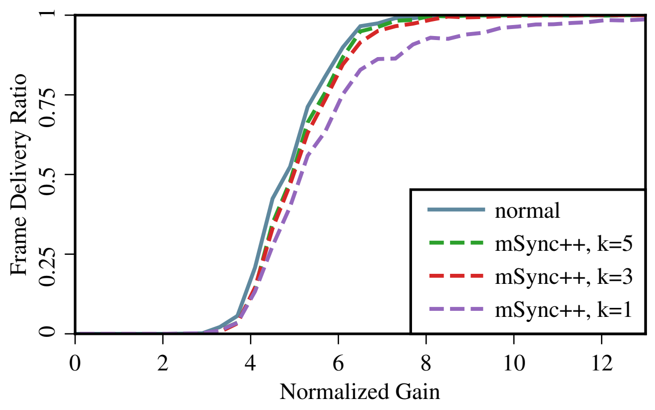
To demonstrate the practical feasibility of the approach, we also implement it on SDR. As discussed earlier, the platform of our BATS transceiver supports only short frame sizes and, therefore, does not face the problem that mSync++ solves. For that reason, we decided to implement our improved algorithm for the IEEE 802.15.4 transceiver, where we can significantly reduce the amount of buffered data. Apart from that, IEEE 802.15.4 is the more complex physical layer, highlighting the general applicability of the idea.

In our proof-of-concept implementation, we varied $k$ to the number of samples corresponding to 1 byte, 3 byte and 5 byte. Even with the largest $k$, we are able to reduce the amount of buffered samples by over 96 % for a full sized frame (from 129 byte to 5 byte). However, the important aspect is that the buffered data is constant and independent from the maximum frame size. This makes mSync++ an interesting option also for other physical layers that support larger frame sizes.

We start our evaluation with simulations over an AWGN channel. To ease comparison, we use 30 byte frames with pseudo-random payload, like in prior experiments. The resulting packet delivery ratio for different SNR levels is



(a) Simulations over an AWGN channel.



(b) Experiments in an office environment.

Figure 13. Observed packet delivery ratio of the IEEE 802.15.4 PHY in simulations and measurements.

depicted in Figure 13a. For the sake of readability, we did not plot the confidence intervals as the lines are very close. The figure is, however, based on the same number of measurements as the previous plots and showed a similar confidence level. The results indicate that already low values for $k$ (i.e., $k = 3$ and $k = 5$) provide very similar performance as the normal receiver, which means we can benefit from shorter frames of mSync++ without suffering from frame loss. Furthermore, the plot shows that a $k$ of only 3 was large enough to have the receiver synchronized with a high probability. Otherwise, some SFDs would have been missed, leading to worse performance compared to normal frames.

To validate the simulations and to rule out any unrealistic simplifications, we also conducted real over-the-air measurements. We ran the same configuration with B210 SDRs in an office environment and varied the transmit gain to set different SNR levels. Using this method, we only know the relative change of the SNR, but not the absolute level. In Figure 13b, we shifted the x-axis to a similar level as in the simulations to ease comparison of the results. Overall, we see that mSync++ offers very similar performance as normal frames, proving the practical feasibility of our approach.

## 8 Computational Complexity

We have already looked into most aspects of mSync. In particular, we showed through simulations and experiments that it allows us to reduce the frame size without degrading physical layer performance. For the transmitter, this directly results in energy savings since we are able to send shorter frames without introducing additional complexity. Therefore, mSync allows to trade-off energy consumption at the transmitter against a more complex receiver. In the following, we try to quantify this complexity.

For that reason, we compare mSync and mSync++ with a baseline transceiver that employs blind estimation for symbol timing recovery and performs frame detection through correlating with the SFD in subsequent stages. Such receiver was, for example, used for the SDR-based ground nodes of the Bats project and the GNU Radio IEEE 802.15.4 physical layer. When evaluating the complexity, an important insight is that mSync does not introduce a new signal processing algorithm, but merely changes the input that is fed to the normal algorithm. Thus, the computational overhead of mSync is twofold: First, we have to save and restore the internal state of the algorithm when switching directions (with mSync++, the algorithm has to restore its internal state before it can continue in forward direction). We believe that this overhead is negligible since it comprises only saving and restoring of a few floating point numbers. Second, parts of the sample stream have to be processed twice by the synchronization algorithm. For mSync, the number of samples depends on the frame size, while for mSync++ the number of samples depends on the placement of the SFD, i.e., the parameter $k$.

To present exemplary results, we prepared a sample stream with 30 byte IEEE 802.15.4 frames, as used in the previous experiments. We used a sample rate of 4 Msps and an inter-frame space of 100 ms, corresponding to ten frames per second. The SNR was set to 30 dB to make sure that all frames are received, i.e., that all frames go through the whole decoding process. For mSync++, we used $k = 5$, i.e., placed the SFD after 5 byte.

The resulting sample stream was loaded into memory and piped into the SDR receiver with its real-time sample rate of 4 Msps. Using GNU Radio's performance counters [37], we monitored the CPU time of each block when running the receiver on an Intel i7-7560U processor. To do this, we developed a custom application that connects to the running flow graph, resets all performance counters, waits for 60 s, and writes CPU times of all block into a file. With this approach, we can perform precise measurements, which are not impacted by the start-up time of the flow graph.

The results of these measurements are depicted in Figure 14, where we plot the CPU time of individual receiver components during the 60 s measurement period. The *Demodulator*, *Filter*, and *Subtract* components are for demodulation and normalization of the signal level before feeding it to the synchronization algorithm. These three components are not affected by mSync and, therefore, show very similar CPU times in all modes.

The most interesting component is the synchronization algorithm, which we labeled *Sync* in the figure. Already in the normal configuration, it is the most demanding component. When switching to mSync, we have to process

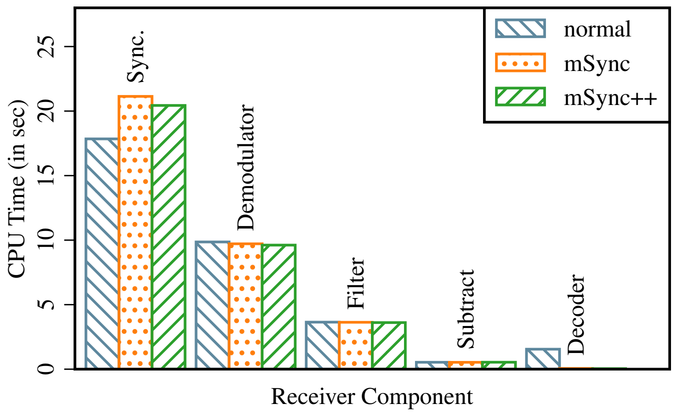


Figure 14. Computational complexity of the IEEE 802.15.4 transceiver running in different modes.

the 30 byte frame twice, which increases the overall CPU time from 17.8 s to 21.1 s. With mSync++, the part that has to be processed twice is reduced to 5 byte, leading to a CPU time of 20.4 s.

Another difference between both mSync variants and the normal receiver is that searching for the SFD becomes part of the synchronization algorithm. Normally, the *Decoder* component processes the continuous bit stream that is output by the synchronization algorithm. In this stream, it searches for the SFD and, once found, decodes the data by demapping the spreading sequences of the IEEE 802.15.4 physical layer to the data bits. With mSync, searching for the SFD becomes part of the synchronization algorithm, since it has to know when to switch directions. That means that the increased CPU times of the mSync variants also stem from the fact that this functionality is shifted from the decoder to the synchronization algorithm. In fact, when using mSync or mSync++, the CPU times of the decoder drop from 1.5 s to below 60 ms, making the values hardly visible in Figure 14.

Overall, these measurements underline the practical feasibility of our approach. While the absolute values might vary depending on the platform, our experiment show that the computational overhead of the mSync variants is manageable.

## 9 Conclusion

We presented a novel physical layer frame format and a corresponding decoding strategy for single carrier wireless communication systems, as often used in Wireless Sensor Networks (WSNs), Cyber-Physical Systems (CPSs), Internet of Things (IoT) devices, and industrial automation systems. Our approach works without dedicated preamble symbols, which results in shorter frames with less physical layer overhead, saving energy and decreasing occupancy of the wireless channel. To assess the performance of our approach and to prove its feasibility, we incorporated it in two different Software Defined Radio (SDR)-based prototypes: a custom Binary Phase Shift Keying (BPSK) ultra low-power transceiver and the Offset Quadrature Phase-Shift Keying (O-QPSK) physical layer of the IEEE 802.15.4 standard. Both simulations and over-the-air measurements showed that omitting the preamble did not degrade physical layer performance. On the contrary, through analytical evaluations, we were able to show that the shorter frames can improve the goodput of networks considerably.

Finally, we addressed the drawback of our algorithm, i.e., the need to buffer samples in the receiver. With a simple variation of the algorithm, we can reduce the buffered data to a small constant number of samples, which makes our approach applicable to physical layers independent from their maximum frame size. We believe that the algorithm occupies a sweet spot between performance and computational complexity, making it an attractive option for a broad range of single carrier communication systems.

## Acknowledgements

This work has been supported in part by the German Research Foundation (DFG) under grant no. FOR 1508.

## References

[1] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A Survey,” *Elsevier Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[3] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, “From wireless sensor networks towards cyber physical systems,” *Elsevier Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 397–413, Aug. 2011.

[4] F. Dressler, B. Bloessl, M. Hierold, C.-Y. Hsieh, T. Nowak, R. Weigel, and A. Koelpin, “Protocol Design for Ultra-Low Power Wake-Up Systems for Tracking Bats in the Wild,” in *IEEE International Conference on Communications (ICC 2015)*. London, UK: IEEE, Jun. 2015, pp. 6345–6350.

[5] C. Rutz, Z. T. Burns, R. James, S. M. Ismar, J. Burt, B. Otis, J. Bowen, and J. J. S. Clair, “Automated mapping of social networks in wild birds,” *Current Biology*, vol. 22, no. 17, pp. R669–R671, 2012.

[6] I. Dietrich and F. Dressler, “On the Lifetime of Wireless Sensor Networks,” *ACM Transactions on Sensor Networks*, vol. 5, no. 1, pp. 1–39, Feb. 2009.

[7] M. Kubisch, H. Karl, and A. Wolisz, “Distributed Algorithms for Transmission Power Control in Wireless Sensor Networks,” in *IEEE Wireless Communications and Networking Conference (WCNC 2003)*, New Orleans, LA, Mar. 2003.

[8] W. Ye, J. Heidemann, and D. Estrin, “An Energy-Efficient MAC Protocol for Wireless Sensor Networks,” in *21st IEEE Conference on Computer Communications (INFOCOM 2002)*. New York, NY: IEEE, Jun. 2002, pp. 1567–1576.

[9] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, “Low power, low delay: Opportunistic routing meets duty cycling,” in *11th ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN 2012)*. Beijing, China: IEEE, Apr. 2012, pp. 185–196.

[10] I. Demirkol, C. Ersoy, and F. Alagoz, “MAC Protocols for Wireless Sensor Networks: a Survey,” *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, Apr. 2006.

[11] B. Bloessl and F. Dressler, “mSync - Frames without Preambles,” in *21st ACM International Conference on Mobile Computing and Networking (MobiCom 2015), 4th ACM Software Radio Implementation Forum (SRIF 2015), Poster Session*. Paris, France: ACM, Sep. 2015, pp. 11–11.

[12] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, “Wireless Sensor Networks: a Survey on the State of the Art and the 802.15.4 and ZigBee Standards,” *Elsevier Computer Communications*, vol. 30, no. 7, pp. 1655–1695, May 2007.

[13] “Low-Rate Wireless Personal Area Networks (LR-WPANs),” IEEE, Std 802.15.4-2011, Jun. 2011.

[14] J. M. Kahn, R. Katz, and K. Pister, “Emerging Challenges: Mobile Networking for ’Smart Dust’,” *Journal of Communications and Networking*, vol. 2, no. 3, Sep. 2000.

[15] M. Stemm, R. H. Katz, and Y. H. Katz, “Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices,” *IEICE Transactions on Communications*, vol. E80-B, no. 8, pp. 1125–1131, Aug. 1997.

[16] 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 and Tutorials*, vol. 15, no. 1, pp. 101–120, Feb. 2013.

[17] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*. Baltimore, MD: ACM, 2004, pp. 95–107.

[18] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, Dec. 2011.

[19] C. Gomez, J. Oller, and J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, Sep. 2012.

[20] I. Demirkol, C. Ersoy, and E. Onur, "Wake-up receivers for wireless sensor networks: benefits and challenges," *IEEE Wireless Communications*, vol. 16, no. 4, pp. 88–96, Aug. 2009.

[21] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, and G. Borriello, "Building the Internet of Things Using RFID," *IEEE Internet Computing*, vol. 33, no. 3, pp. 48–55, May/June 2009.

[22] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient Backscatter: Wireless Communication Out of Thin Air," in *ACM SIGCOMM 2013*. Hong Kong, China: ACM, Aug. 2013, pp. 39–50.

[23] J. Kimionis, A. Bletsas, and J. N. Sahalos, "Increased Range Bistatic Scatter Radio," *IEEE Transactions on Communications*, vol. 62, no. 3, pp. 1091–1104, Mar. 2014.

[24] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive Wi-Fi: Bringing Low Power to Wi-Fi Transmissions," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2016)*. Santa Clara, CA: USENIX, Mar. 2016, pp. 151–164.

[25] F. Harris, "Let's Assume the System Is Synchronized," in *Globalization of Mobile and Wireless Communications*, ser. Signals and Communication Technology, R. Prasad, S. Dixit, R. van Nee, and T. Ojanpera, Eds. Springer, 2011, pp. 311–325.

[26] F. J. Harris and M. Rice, "Multirate Digital Filters for Symbol Timing Synchronization in Software Defined Radios," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 12, pp. 2346–2357, Dec. 2001.

[27] K. Mueller and M. Müller, "Timing Recovery in Digital Synchronous Data Receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 516–531, May 1976.

[28] T. W. Rondeau, "On the GNU Radio Ecosystem," in *Opportunistic Spectrum Sharing and White Space Access: The Practical Reality*, O. Holland, H. Bogucka, and A. Medeisis, Eds. Wiley, May 2015, pp. 25–48.

[29] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly, "WARP: A Flexible Platform for Clean-Slate Wireless Medium Access Protocol Design," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 12, no. 1, pp. 56–58, Jan. 2008.

[30] G. Sklivanitis, A. Gannon, S. N. Batalama, and D. A. Pados, "Addressing Next-Generation Wireless Challenges with Commercial Software-Defined Radio Platforms," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 59–67, Jan. 2016.

[31] F. Dressler, S. Ripperger, M. Hierold, T. Nowak, C. Eibel, B. Cassens, F. Mayer, K. Meyer-Wegener, and A. Koelpin, "From Radio Telemetry to Ultra-Low-Power Sensor Networks: Tracking Bats in the Wild," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 129–135, Jan. 2016.

[32] M. Nabeel, B. Bloessl, and F. Dressler, "On Using BOC Modulation in Ultra-Low Power Sensor Networks for Wildlife Tracking," in *IEEE Wireless Communications and Networking Conference (WCNC 2016)*. Doha, Qatar: IEEE, Apr. 2016, pp. 848–853.

[33] T. Schmid, "GNU Radio 802.15.4 En-and Decoding," Networked & Embedded Systems Laboratory, UCLA, Technical Report TR-UCLA-NESL-200609-06, Jun. 2006.

[34] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer, "A GNU Radio-based IEEE 802.15.4 Testbed," in *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, Cottbus, Germany, Sep. 2013, pp. 37–40.

[35] S. Pollin, M. Ergen, S. C. Ergen, B. Bougard, L. Van der Perre, I. Moerman, A. Bahai, P. Varaiya, and F. Catthoor, "Performance Analysis of Slotted Carrier Sense IEEE 802.15.4 Medium Access Layer," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3359–3371, Sep. 2008.

[36] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, Mar. 2000.

[37] T. W. Rondeau, T. O'Shea, and N. Goergen, "Inspecting GNU Radio Applications with ControlPort and Performance Counters," in *ACM SIGCOMM 2013, 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum (SRIF 2013)*. Hong Kong, China: ACM, Aug 2013, pp. 65–70.

**Bastian Bloessl** received the diploma degree in Computer Science from the University of Würzburg, Germany, in 2011. He is working toward the PhD degree at the chair for Distributed Embedded Systems at the Heinz Nixdorf Institute and the Dept. of Computer Science, Paderborn University. In 2015, he won a FitWeltweit scholarship from the German Academic Exchange Service (DAAD), which funded a six-month stay in the research group of Prof. Mario Gerla at the Computer Science Department of the University of California, Los Angeles (UCLA). Since 2017, he has been a researcher at the CONNECT Center, Trinity College Dublin, Ireland's Research Center for Future Networks and Communications, where he is funded through a Marie Skłodowska-Curie fellowship. His research is focused on using software defined radio-based prototypes to assess the performance and robustness of vehicular and sensor networks. He is a member of the IEEE.

**Falko Dressler** received the MSc and PhD degrees from the Department of Computer Science, University of Erlangen, in 1998 and 2003, respectively. He is full professor of computer science and chair for Distributed Embedded Systems at the Heinz Nixdorf Institute and the Dept. of Computer Science, Paderborn University, where he is also a member of the University Senate. He is associate editor-in-chief for Elsevier Computer Communications as well as an editor for journals such as the IEEE Transaction on Mobile Computing, the IEEE Transaction on Network Science and Engineering, the Elsevier Ad Hoc Networks, and the Elsevier Nano Communication Networks. He has been guest editor of special issues in the IEEE Journal on Selected Areas in Communications, the IEEE Communications Magazine, the Elsevier Ad Hoc Networks, and many others. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM, and many others. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. He is a fellow the IEEE as well as a senior member of the ACM, and member of the GI (German Computer Science Society). He also serves on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include adaptive wireless networking, self-organization techniques, and embedded system design with applications in ad hoc and sensor networks, vehicular networks, industrial wireless networks, and nano-networking.