Synchronous Transmissions in Low-Power Wireless: A Survey of Communication Protocols and Network Services

MARCO ZIMMERLING, TU Dresden, Germany
LUCA MOTTOLA, Politecnico di Milano, Italy and RISE, Sweden
SILVIA SANTINI, Universitá della Svizzera italiana (USI), Switzerland

Low-power wireless communication is a central building block of Cyber-physical Systems and the Internet of Things. Conventional low-power wireless protocols make avoiding packet collisions a cornerstone design choice. The concept of synchronous transmissions challenges this view. As collisions are not necessarily destructive, under specific circumstances, commodity low-power wireless radios are often able to receive useful information even in the presence of superimposed signals from different transmitters. We survey the growing number of protocols that exploit synchronous transmissions for higher robustness and efficiency as well as unprecedented functionality and versatility compared to conventional designs. The illustration of protocols based on synchronous transmissions is cast in a conceptional framework we establish, with the goal of highlighting differences and similarities among the proposed solutions. We conclude the paper with a discussion on open research questions in this field.

CCS Concepts: • **Networks** → **Network protocol design**; **Link-layer protocols**; **Network layer protocols**; **Network services**; • **Computer systems organization** → *Sensor networks*; *Sensors and actuators*; *Embedded systems*; *Reliability*; *Redundancy*; *Fault-tolerant network topologies.*

Additional Key Words and Phrases: Low-power wireless networks, synchronous transmissions, capture effect, message-in-message effect, constructive interference, sender diversity, simplicity, multi-hop communication

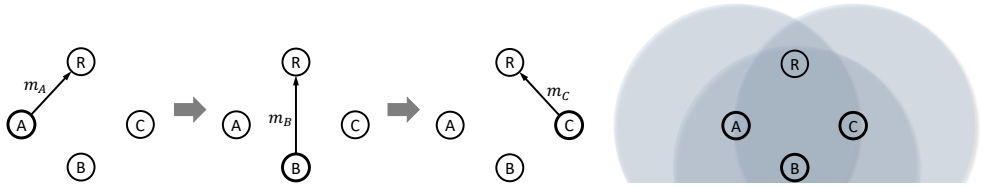
ACM Reference Format:
Marco Zimmerling, Luca Mottola, and Silvia Santini. 2020. Synchronous Transmissions in Low-Power Wireless: A Survey of Communication Protocols and Network Services. *ACM Comput. Surv.* 1, 1, Article 1 (January 2020), 37 pages. **https://doi.org/10.1145/3410159**

1 INTRODUCTION

Cyber-physical Systems (CPS) and the Internet of Things (IoT) blur the boundary between the physical and cyber domains, enabling powerful applications in areas as diverse as industrial automation, smart health, and ambient intelligence [73, 119]. Their concrete realization relies on embedded hardware and software with distinctive features. Crucially, to enable pervasive and untethered deployments, the devices communicate wirelessly and are powered by batteries or small capacitors that store limited amounts of energy, possibly harvested from the environment [10].

Authors' addresses: Marco Zimmerling, TU Dresden, Center for Advancing Electronics Dresden (CfAED), Networked Embedded Systems Lab, Helmholtzstrasse 18, BAR II59 (D wing, second floor), Dresden, 01187, Germany, **marco.zimmerling@tu-dresden.de**; Luca Mottola, Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Networked Embedded Software Lab, Via Golgi 42, Building 22 (room 319, third floor), Milan, 20133, Italy, RISE, Digital Systems Division, Connected Intelligence, Electrum, Isafjordsgatan 22/Kistagången 16 (sixth floor), Kista, 16440, Sweden, **luca.mottola@polimi.it**; Silvia Santini, Universitá della Svizzera italiana (USI), Faculty of Informatics, Via Buffi 13, Informatics Building (office 108, level 1), Lugano, 6900, Switzerland, **silvia.santini@usi.ch**.

0360-0300/2020/1-ART1 $15.00
https://doi.org/10.1145/3410159



(a) Using link-based transmissions nodes A, B, and C transmit one after another to avoid collisions at the receiver R.

(b) Using synchronous transmissions all nodes transmit at the same time.

Fig. 1. Illustration of link-based transmissions in (a) versus synchronous transmissions in (b).

Fundamental design trade-offs are pursued to stay within the small energy budgets. Low-complexity physical layers enable low-power wireless transceiver implementations, but this comes at the cost of low data rates (tens of kbit/s to a few Mbit/s) and limited communication ranges (tens of meters to a few kilometers) depending on the technology. IEEE 802.15.4 and Bluetooth Low Energy (BLE) are examples of short-range low-power wireless technologies, whereas LoRa and Sigfox belong to the corresponding class of long-range technologies. In addition, the radios and other device components are duty cycled by switching them on and off based on application requirements and communication demands to reduce energy consumption; however, a device cannot receive any packet while its radio is off.

The goal of minimizing energy consumption is thus at odds with other key performance objectives, such as increasing the fraction of packets delivered to the destinations across the unreliable wireless channel, which impacts *reliability*, or reducing the time needed to do so, which affects *latency*. Because of limited communication ranges, multi-hop communication is often employed whereby intermediate nodes relay packets from sources to destinations, which further compounds the problem. Since the early 2000's, the tension among those conflicting goals has motivated a profusion of research efforts in low-power wireless communication and networking [2, 19, 25].

Link-based and synchronous transmissions. Low-power wireless communication is broadcast in nature: the radio waves emitted by a transmitter travel through the air and can be received by any device in communication range and with the appropriate equipment. If a device is in the communication range of multiple transmitters, that device sees the superposition of multiple signals whenever two or more transmitters transmit at the same time on the same carrier frequency. The common belief suggests that such *packet collisions* are destructive and prevent a device from receiving any useful information. To achieve high reliability and energy efficiency, conventional low-power wireless protocols try to avoid collisions using techniques such as carrier sensing, handshaking, and scheduling. As a result, neighboring devices transmit packets one after another over *point-to-point links*, as illustrated in Fig. 1a. We refer to this concept as *link-based transmissions*, which are used by the vast majority of low-power wireless protocols, for example, as it allows to adopt well-known approaches from wired networking such as routing.

Since 2010, a new breed of low-power wireless communication protocols emerged based on the concept of *synchronous transmissions*. Unlike link-based transmissions, the tenet of synchronous transmissions is that collisions are not necessarily destructive. Rather, by purposely letting multiple nodes transmit at the same time on the same carrier frequency, as shown in Fig. 1b, with high probability a node can receive useful information nonetheless. This allows protocols using synchronous transmission to embrace the broadcast nature of low-power wireless communication instead of hiding it under the artificial abstraction of point-to-point links.

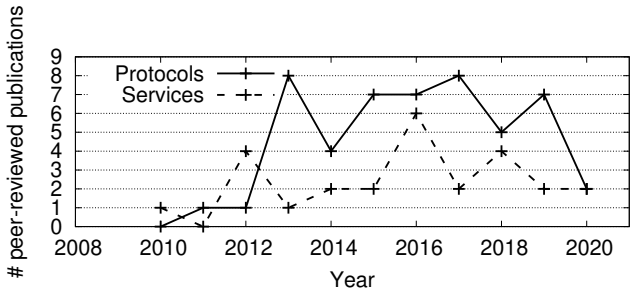


Fig. 2. Number of peer-reviewed conference and journal publications over time as surveyed in this article, covering the design of communication protocols and network services based on synchronous transmissions.

There exist related concepts in wireless communications, including cooperative diversity [72, 115, 116] and non-orthogonal multiple access [109] based on, for example, successive interference cancellation [134]. While synchronous transmissions can be considered a specific variant of those concepts, a key difference is that synchronous transmissions are applicable to off-the-shelf commodity low-power wireless devices and existing physical layers, including IEEE 802.15.4, BLE, and LoRa. Thus, millions of cheap microcontroller-based devices deployed around the world support synchronous transmissions out of the box.

The gains and benefits of synchronous transmissions over link-based transmissions range from orders of magnitude lower latencies [29] and energy consumption [52] to providing the ability functionality such as virtual synchrony [40] that was previously thought unfeasible [120]. Researchers from academia and industry also built on the concept of synchronous transmissions when participating to international scientific competitions [114], demonstrating, for example, the robustness of this technique even under severe interference from co-located networks. Synchronous transmissions thereby offer protocol designers a radically different perspective and unprecedented possibilities. Nevertheless, compared to traditional protocols built on link-based transmissions, the use of synchronous transmissions is often based on different assumptions, requires modified protocol layering and architectures, and demands alternative approaches to evaluating performance.

Fig. 2 shows the take-up of synchronous transmissions over time for the design of communication protocols and network services in the peer-reviewed publications we include in this survey, demonstrating the growing interest. Following pioneering work that appeared between 2010 and 2012, the bulk of the technical and scientific contributions is arguably found in the last five to six years. As the body of published material grows, it is now time to consolidate the knowledge and to reflect on what are the challenges and opportunities yet to be explored. A recent tutorial [20] makes a first attempt toward this goal, but covers only a small fraction of the published material and does not present a conceptual framework that elicits the similarities and differences among existing solutions.

Contribution and road-map. This paper presents an extensive analysis of the use of synchronous transmissions in the design of low-power wireless communication protocols and network services. Coherently surveying the topic requires the necessary background for readers to build upon. Sec. 2 provides a primer on synchronous transmissions in low-power wireless communications, focusing on the conditions and effects that enable a successful packet reception in the presence of collisions and the resulting benefits. Readers interested in a more detailed treatment of the underlying physical-layer principles are referred to our complementary paper [6], which also surveys works seeking to understand and improve the reliability of the synchronous transmission technique itself.

Sec. 3 represents the core of this paper. It contributes an extensive analysis and classification of existing low-power wireless communication protocols employing synchronous transmissions. The semantics we associate to this term defines the scope of our work: In low-power wireless networks, a *communication protocol* is regarded as providing applications with functionality for data collection, data dissemination, or peer-to-peer interactions [34, 50, 60, 70, 78, 93, 96, 97]. Therefore, a communication protocol enables the exchange of *user-defined data* among physical devices, therefore offering an even minimal programming interface that allows *applications* to access its functionality. Our review of existing works in this area unfolds through the analysis of key dimensions that characterize the state of the art, providing a novel conceptual framework that serves as guidance through the literature. For each dimension of this framework, we first discuss the existing individual instances and then describe an existing protocol as a paradigmatic example.

There also exists a number of solutions that use synchronous transmissions to realize network primitives not expressly meant to transport application data, but to provide higher-level network functionality (*e.g.*, consensus) or auxiliary network services (*e.g.*, time synchronization). We discuss those solutions in Sec. 4. Then, in Sec. 5, we outline a variety of open challenges and research questions. These issues pertain to fields as diverse as understanding and exploiting the peculiar features of synchronous transmissions in the design of communication protocols, conceiving novel network stack designs to better leverage synchronous transmissions, the alternative definition of communication patterns and network services, the integration of low-power wireless systems using synchronous transmissions with standard network infrastructures, and standardization processes. Many of these challenges require expertise from diverse fields that hitherto experience little cross-fertilization. This paper is also the opportunity to foster such an exchange of knowledge and insights between the relevant disciplines. We end the paper in Sec. 6 with brief concluding remarks.

2 A PRIMER ON SYNCHRONOUS TRANSMISSIONS IN LOW-POWER WIRELESS

This section provides a brief overview of the physical-layer principles that enable synchronous transmissions on off-the-shelf low-power wireless platforms, and works out the key advantages of synchronous transmissions especially from a protocol design perspective.

2.1 Receiving in the Presence of Packet Collisions

Understanding under what circumstances a node can successfully receive a packet in the presence of collisions motivated a wealth of research [80, 141]. Current knowledge indicates that two effects play an important role, often in combination, depending on the low-power receiver implementation and on the properties of the incoming signals: capture effect [75] and constructive interference [41]. We briefly discuss these physical-layer effects in the following, focusing on the most relevant aspects needed as background for the remainder of this article, while Sec. 5 outlines future research directions including implications of a third physical-layer effect known as message-in-message [91] on protocol design. For a more comprehensive description of these physical-layer phenomena we refer the reader to our complementary paper [6].

Capture effect. The capture effect enables a receiver to correctly receive one of the synchronously transmitted packets if the incoming signals satisfy certain power and timing conditions. Specifically, the difference in received power between one signal and the sum of all other signals plus noise must exceed the *capture threshold*. If such strongest signal arrives first, the receiver will correctly receive it with high probability, while treating all other signals as noise. Otherwise, if such strongest signal arrives after a weaker signal, it must do so within the *capture window*, that is, before the receiver finishes receiving the preamble of the earlier packet. The reason is that, after receiving the first valid preamble, low-power wireless radios suspend preamble search until they have received the

complete packet. Thus, the radio cannot "switch" to a stronger signal if it arrives too late, resulting in an erroneous reception as the stronger signal destroys the weaker signal.

The capture threshold and capture window depend on the physical layer. For example, using IEEE 802.15.4 radios with O-QPSK modulation in the 2.4 GHz band, experimental studies report a capture threshold of 2-3 dB [31, 69, 80] and a capture window of 128 μs [145]. The capture effect is a general phenomenon already exploited in a variety of other popular wireless technologies, such as BLE [107], Bluetooth [23], IEEE 802.11 variants [74], LoRa [14], and sub-GHz radios [140].

Constructive interference. If all signals arrive with similar power, a successful reception is only possible if the transmitted packets are identical and so-called *constructive interference* occurs [41]. True constructive interference means that the signals perfectly overlap, that is, there are no time and phase offsets. However, even if transmitters can compensate for different path delays, and thus make the signals arrive without time and phase offsets at the receiver, the phase offsets would vary *throughout* the reception because of different carrier frequencies across the transmitters.

Carrier frequency offsets are inevitable without a shared clock: the radios' oscillators generating the carriers run asynchronously at different frequencies within a certain tolerance range. For example, according to the IEEE 802.15.4 standard, the oscillators are allowed to drift by up to ±40 ppm, which translates into a maximum carrier frequency offset of 192 kHz between sender and receiver for the 2.4 GHz band. Thus, in a real network of commodity low-power wireless devices, true constructive interference is highly improbable. Rather, studies reporting higher median and variance in received power suggest that, as a result of the carrier frequency offsets, there are alternating periods of constructive and destructive interference during a reception [33, 69, 80, 138].

Periods of destructive interference can cause bit errors at the receiver, and without additional measures those lead to an erroneous packet reception. One possible measure is a channel code that enables a receiver to correct a certain number of bit errors. For example, the direct-sequence spread spectrum (DSSS) technique used by 2.4 GHz IEEE 802.15.4 radios tolerates twelve incorrect bits (called chips) per symbol. With this error-correction capability, a receiver is able to successfully receive an entire packet despite a range of possible carrier frequency offsets if the time offset among the signals is less than 0.5 μs [80]. The ability to cope with carrier frequency offsets is strongly related to the receiver implementation, especially whether the demodulator works in a coherent or non-coherent fashion (*e.g.*, O-QPSK or MSK in case of 2.4 GHz IEEE 802.15.4) [38]. This may explain empirical results indicating that synchronous transmissions work even in the absence of a suitable error-correcting mechanism [4], such as most BLE variants or certain sub-GHz radios.

In summary, although the term constructive interference is technically incorrect and somewhat misleading, it nonetheless describes the fact that it *appears* to a link-layer protocol as if the transmissions interfere constructively, whereas the underlying carrier signals do not. The impact of physical-layer parameters, receiver implementations, and error-correction mechanisms on the occurrence of constructive interference is an important topic for future research (see Sec. 5).

2.2 Benefits of Synchronous Transmissions

As discussed, synchronous transmissions only work under certain conditions. It is not obvious whether it is easy or difficult to meet those conditions, and if doing so can improve the communication performance and reliability compared to link-based transmissions. To illustrate this, let us assume that the link from node A to the receiver R in Fig. 1a is of excellent quality, that is, it has a packet reception ratio (PRR) very close to 100 % when using link-based transmissions. In this case, it is extremely unlikely that the PRR at the receiver R improves when nodes B and C transmit synchronously with node A. It is much more likely that the PRR with synchronous transmissions slightly degrades because a certain chance exists that none of the above physical-layer effects

occurs [80], and a packet collision rather happens that prevents any packet from being correctly received. Although real-world experiments show that synchronous transmissions improve PRR for low- and medium-quality links [135], this alone does not explain the increasing popularity of synchronous transmissions in low-power wireless systems.

The works reviewed in this and our complementary paper [6] demonstrate that creating the conditions under which synchronous transmissions work in real-world low-power power wireless networks is possible with no or only little overhead in a wide range of realistic settings. Synchronous transmissions are thereby demonstrated to provide key benefits over link-based transmissions:

(1) *Sender diversity.* Compared to link-based transmissions, the presence of multiple transmitters allows a receiver to benefit from *sender diversity*. If the transmitters are at least one carrier wavelength apart (*e.g.*, about 0.125 m at 2.4 GHz), their wireless links to the receiver are uncorrelated with high probability [132]. It is therefore unlikely that all links suffer fading or other adverse effects at the same time. This increases the chances of successful reception at a common receiver, compared to the expected reliability of link-based transmissions in comparable conditions and unless the pair-wise links are already near-perfect.
(2) *Simplicity.* The observation that packet collisions are not necessarily harmful questions the need for information and mechanisms to avoid them. This includes the collection of information about the network state, such as a node's neighbors and the corresponding link qualities, as well as mechanisms like link estimators, link schedulers, and carrier sense multiple access (CSMA). These and other mechanisms are used extensively by conventional low-power wireless protocols to enable efficient and reliable multi-hop communication. The opportunity to drop them enables dramatically simpler protocol designs.

These are the only benefits provided by the concept of synchronous transmissions *itself*. They are, however, fundamental in that they offer protocol designers a radically different perspective and unprecedented possibilities compared to link-based transmissions. The extent to which these possibilities and benefits are exploited and possibly combined with other higher-level mechanisms depends on the (multi-hop) *protocol design*; existing solutions show that the potential end-to-end gains are diverse and enormous. For example, the degree of sender diversity is a function of the node density and the number of active synchronous transmitters, and can be combined with other forms of diversity including receiver diversity, obtained in settings with more than one receiver, frequency diversity, when nodes use channel hopping, and temporal diversity, when a message is transmitted multiple times and possibly by different sets of synchronous transmitters.

Based on this, multi-hop communication protocols have been designed whose logic is independent of the network state and whose performance can be accurately modeled by simple discrete-time Markov chains [39, 152]. On the other hand, nothing prevents protocol designers from leveraging long-lived network state for optimizing performance in specific scenarios, possibly at the cost of reduced robustness to network dynamics. The following two sections analyze the many ways in which synchronous transmissions have been used in the design of low-power wireless communication protocols and network services.

3 SYNCHRONOUS TRANSMISSIONS IN COMMUNICATION PROTOCOLS

This section analyzes existing multi-hop communication protocols built on synchronous transmissions. To this end, we first identify key protocol characteristics allowing us to separate out the fundamental features of a protocol's design, independent of its concrete implementation. Fig. 3 depicts the taxonomy resulting from this analysis, which also serves as road-map and reference throughout this section. For every dimension of the taxonomy, we describe one protocol in more detail to exemplify a given feature, and briefly comment on other protocols whenever these build on

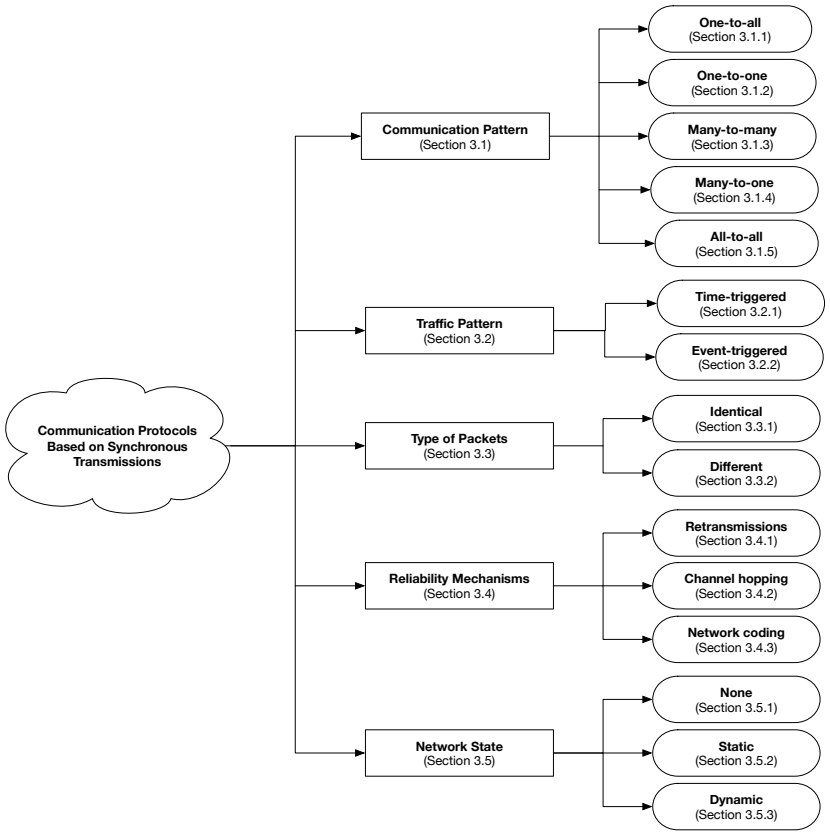


Fig. 3. Taxonomy and road-map for classifying communication protocols built on synchronous transmissions.

top of the primary example or exhibit further interesting characteristics. Table 1 lists all protocols surveyed in this section and indicates their classification according to taxonomy in Fig. 3.

Existing protocols using synchronous transmissions are extremely diverse in terms of performance goals, design features, and implementation. This diversity is also often reflected in different evaluation methodologies used in the original papers, ranging from, for example, simulations and table-top experiments to large-scale experiments on public testbeds. It is therefore impossible to *quantitatively* compare existing communication protocols in a fair way based on performance results reported in the original papers. For this reason, we focus on providing a *qualitative* comparison, and identify rigorous benchmarking as an important direction for future work (see Sec. 5).

3.1 Communication Pattern

We begin by considering the communication patterns supported by existing protocols. With this we refer to the *intent* of the protocol, that is, the communication patterns offered to higher-level protocols or applications, and not to the protocol implementation. We distinguish protocols that

- support exactly *one* sender;
- support any number of senders between one and all, referred to as *many*;
- support only scenarios in which *all* nodes in the network operate as senders.

The number of receivers, on the other hand, indicates the number of distinct physical devices that the protocol *intends* to deliver data to. We distinguish protocols that transmit data to

Table 1. Overview of synchronous transmission based communication protocols surveyed in Sec. 3, classified according to the taxonomy in Fig. 3.

Protocol	Commun. pattern	Traffic pattern	Type of packets	Reliability mechanisms	Network state
Glossy [41]	**One-to-all**	Time-triggered	Identical	RTX (local)	None
SCIF [136, 137]	One-to-all	Time-triggered	Identical	RTX (local)	**Static**
Splash [27]	One-to-all	Time-triggered	Identical	**Channel hopping**, RTX (local, e2e), network coding	Static
Ripple [146]	One-to-all	Time-triggered	Identical	Network coding, channel hopping	Dynamic
Zippy [123]	One-to-all	**Event-triggered**	Identical	RTX (local)	None
Pando [28, 29]	One-to-all	Time-triggered	Identical	**Network coding**, channel hopping	Static
RedFixHop [37]	One-to-all	Time-triggered	Identical	RTX (local)	None
LiM [150]	One-to-all	Time-triggered	Identical	RTX (local)	Dynamic
BlueFlood [4]	One-to-all	Time-triggered	Identical	Channel hopping, RTX (local)	**None**
Whisper [15]	One-to-all	Time-triggered	Identical	RTX (local)	Dynamic
CXFS [18]	**One-to-one**	Time-triggered	Identical	RTX (local)	Dynamic
Sparkle [147]	One-to-one	Time-triggered	Identical	RTX (local)	Dynamic
PEASST [61]	One-to-one	Time-triggered	**Identical**	RTX (local)	Dynamic
P^3 [26]	One-to-one	Time-triggered	Identical	Channel hopping, RTX (e2e)	**Dynamic**
RFT [149]	One-to-one	Time-triggered	Identical	RTX (local)	Dynamic
LaneFlood [16]	One-to-one	**Time-triggered**	Identical	RTX (local)	Dynamic
LWB [39]	**Many-to-many**	Time-triggered	Identical	RTX (local)	None
eLWB [126]	Many-to-many	Event-triggered	Identical	RTX (local)	None
BLITZ [124, 125]	Many-to-many	Event-triggered	Identical	RTX (local)	None
WTSP [128]	Many-to-many	Time-triggered	Identical	RTX (local, e2e)	None
Blink [154]	Many-to-many	Time-triggered	Identical	RTX (local)	None
Codecast [94]	Many-to-many	Time-triggered	Different	Network coding, RTX (local)	Dynamic
Mixer [47, 90]	Many-to-many	Time-triggered	**Different**	Network coding, RTX (local)	Dynamic
ByteCast [108]	Many-to-many	Time-triggered	Different	RTX (local)	Dynamic
Crystal [52, 53]	**Many-to-one**	Time-triggered	Different	Channel hopping, RTX (local, e2e)	Dynamic
Zimmerling et al. [152]	Many-to-one	Time-triggered	Identical	**RTX** (local, e2e)	None
Choco [129]	Many-to-one	Time-triggered	Identical	RTX (local, e2e)	None
Sleeping Beauty [112]	Many-to-one	Time-triggered	Identical	RTX (local)	Dynamic
FLEET [111]	Many-to-one	Time-triggered	Identical	RTX (local)	Dynamic
Harmony [87]	Many-to-one	Time-triggered	Identical	Channel hopping, network coding, RTX (local)	Dynamic
PiP [88]	Many-to-one	Time-triggered	Different	RTX (local)	Dynamic
Chaos [71]	**All-to-all**	Time-triggered	Different	RTX (local)	None

Protocols marked in bold are those described as the representative example of the corresponding dimension in our taxonomy. Abbreviations: e2e – end-to-end, RTX – retransmissions.

- exactly *one* node in the network;
- any number of nodes between one and all, referred to as *many*;
- *all* nodes in the network.

The above distinction results in nine different classes. However, as we shall see below, concrete instances of protocols found in the literature only fall into a subset of these classes.

3.1.1 One-to-all. Approaches that enable a sender to transmit packets to all other nodes in the network belong to this class; relevant examples include network flooding protocols that send single packets and data dissemination protocols that send multiple packets forming a larger data object.

Example → Glossy: An example of a *one-to-all* communication protocol that uses synchronous transmissions is Glossy [41]. Glossy provides network flooding and optionally time synchronization. The work has played a seminal role in the evolution of research in this area as many communication protocols and network services build on or take inspiration from it.

The single sender in Glossy is called *initiator*. It starts the flooding process by transmitting the packet, while all other nodes in the network have their radio in receive mode. Provided propagation delays are negligible, the one-hop neighbors of the initiator receive the packet all at the same time. Glossy then makes these nodes retransmit the received packet so that the retransmission delay is constant and equal for all involved devices. This way, the one-hop neighbors of the initiator that received the packet retransmit it at nearly the same time. Because these nodes are all sending the same packet, constructive interference can occur in the way described in Sec. 2. This enables the two-hop neighbors of the initiator to receive and retransmit the packet, and so on.

To ensure a constant retransmission delay, Glossy executes a minimal, deterministic sequence of operations between packet reception and packet retransmission. State-of-the-art radios offer features allowing to completely avoid this software processing. Further, Glossy floods are confined to reserved time slots, where no other tasks (application, operating system, etc.) are allowed to execute on a node. This avoids interference on shared resources, which would cause unpredictable jitter. Due to its operation, Glossy also enables nodes to time-synchronize with the initiator.

Without retransmission delay, Glossy achieves the theoretical lower latency bound for one-to-all communication of a single packet in multi-hop networks using half-duplex radios. By retransmitting the packet a configurable number of times during a flood, Glossy exploits different forms of diversity for high reliability. The diversity also de-correlates packet receptions and losses [63, 152], which simplifies modeling and the design of higher-level functionality such as feedback controllers [89]. Nodes do not maintain any network state, which makes Glossy highly robust to network dynamics resulting from moving or failing nodes, interference from co-located wireless networks, etc.

Completing the picture. The Spine Constructive Interference-based Flooding (SCIF) protocol [136, 137] extends Glossy with a topology-control mechanism that makes it scale to high-density or large-scale networks, where the authors report that the performance of synchronous transmissions degrades. RedFixHop [37] uses the same transmission technique as Glossy, but encodes the payload in the Medium Access Control (MAC) packet sequence number. Upon the reception of a packet, a node forwards it using hardware-generated acknowledgments. Because of the deterministic operation of the radio hardware, the retransmission delay is further reduced and better aligned across senders. Less is More (LiM) [150] uses reinforcement learning to reduce the number of transmissions during a Glossy flood for improved energy efficiency, while maintaining high flooding reliability. Recent work shows that synchronous transmissions can also be exploited to build Glossy-like protocols on top of low-power wireless physical layers other than IEEE 802.15.4, including ultra-wide band (UWB) [86], BLE [4], and LoRa [79]. For instance, BlueFlood [4] demonstrates one-to-all communication using synchronous transmissions over Bluetooth 5's physical layer.

Unlike Glossy and taking inspiration from [81], a node in BlueFlood performs N back-to-back synchronous transmissions after receiving a packet for increased efficiency, and switches channel after each transmission and reception for increased reliability. Whisper [15] also exploits the idea of back-to-back synchronous transmissions to enable the flooding of small values into the network. However, nodes in Whisper send only a single large packet that contains in the data field up to 18 replicas, called *packlets*, of the actual packet to be sent. Nodes involved in the flood wait for an incoming flood, synchronize their transmissions with the incoming transmission, and then send as many packlets as needed to propagate the flood onward.

While Glossy floods a single packet, a number of dissemination protocols have been designed to distribute multiple packets forming a larger data object. The protocols mainly differ in the way they use pipelined Glossy floods over multiple channels with different forms of packet coding to disseminate all packets quickly and reliably. Splash [27] uses XOR coding and Pando [28, 29] uses Fountain coding, but both protocols assign channels to nodes based on their hop distance from the sender. Ripple [146], instead, assigns channels to packets and uses an erasure code. These protocols rely on information about the network topology (*e.g.*, for channel assignment), which makes them suitable for scenarios in which the nodes are stationary throughout the dissemination process. Quantitative comparison against traditional asynchronous dissemination protocols, such as Deluge, demonstrates a reduction in data dissemination time by a factor of more than 20 [27].

3.1.2 One-to-one. One-to-one protocols enable communication between a specific sender-receiver pair in a multi-hop network. This type of communication primitive is beneficial if the information contained in a message is relevant for—or should be received by—only one destination node.

Example → CXFS: One of the first synchronous transmission based protocols that specifically target one-to-one communication is Concurrent Transmissions with Forwarder Selection (CXFS) [18]. Before the data exchange starts, CXFS performs a two-way handshake between the sender and the intended receiver. During this handshake, the nodes determine whether they are forwarders on a shortest path between the source and the destination. These nodes are included in the *forwarder set* and remain active to relay messages between the sender and the receiver, whereas all other nodes switch their radios off to save energy. Using a tunable parameter, the *boundary b*, the forwarder set can be enlarged to make the data exchange more reliable at the cost of higher energy consumption.

While in Glossy a packet is retransmitted immediately upon reception, CXFS uses a dedicated schedule to organize packet transmissions. In particular, every one-to-one data exchange occurs in time *slots* of fixed length and a given number of *frames* are included in each time slot. At the beginning of the first frame in a slot, the node assigned to that slot can initiate the transmission of a packet. Nodes receiving this packet will retransmit it at the beginning of the next frame, and so on. This way of implementing synchronous transmissions is possible if the time synchronization among nodes is sufficiently accurate to guarantee a precise alignment of frames and slots.

Completing the picture. LaneFlood [16] is similar to CXFS, but operates directly on top of Glossy and lets nodes join the forwarder set with a given probability (refer to Sec. 3.2.1 for a more detailed description). In a similar vein, the Practical Packet Pipeline (P^3) protocol [26] supports the one-to-one transmission of multiple messages (*i.e.*, bulk traffic) along node-disjoint paths determined at a base station, based on global network-topology information collected beforehand (see Sec. 3.5.3 for details). Sparkle [147] also targets one-to-one communication by carefully selecting subsets of nodes that participate in Glossy-like floods. It uses a dedicated transmission-power and topology-control mechanism that seeks to boost the performance of synchronous transmissions. The Reinforcement (RFT) protocol [149] builds upon Sparkle to further improve reliability and energy efficiency, using a time-division multiple access (TDMA) approach for scheduling individual messages.

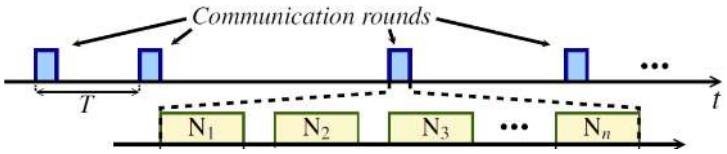


Fig. 4. Many-to-many communication in LWB (taken from [39]). *LWB structures communication into rounds. A round contains a series of slots,* $N_1, \ldots, N_n$*, each of which serves to transmit one message using a Glossy flood.*

3.1.3 Many-to-many. Protocols in this category are capable of disseminating messages from any subset of nodes in the network (senders) to any other desired subset of nodes (receivers). Many-to-many communication is the most universal pattern and essential for many applications and system services, including data replication (*e.g.*, for increased fault tolerance [113]), certain programming abstractions [95], and distributed control tasks (*e.g.*, to coordinate a swarm of aerial drones [103]).

Example → LWB: The Low-power Wireless Bus (LWB) [39] provides to a higher-layer protocol and the application the abstraction of a shared bus, wherein each node can potentially receive all transmitted messages. As shown in Fig. 4, LWB organizes communication into *rounds*, which occur with a variable *period* T. A communication round contains up to n *slots*. Every slot serves to transmit one message from a sender to all other nodes in the network using a Glossy flood; messages are then filtered at the receiver side and passed to a higher-layer protocol or the application only if the node is one of the intended receivers, as specified in the message header. That is, LWB realizes many-to-many communication through a series of one-to-all transmissions followed by message filtering. If the number of messages to be sent exceeds the maximum number of slots n in a round, the many-to-many communication extends across multiple consecutive rounds.

When a node intends to send or receive messages, it first needs to notify LWB's central coordinator, called *host*. To this end, a node sends a *stream request* in a *contention slot*, which is a dedicated slot that is not assigned to a specific sender. For prospective senders the stream request contains, among others, the inter-packet interval with which the stream generates new packets. A node can request multiple streams and dynamically modify or remove a stream by sending a corresponding request to the host. At the end of each round, the host computes the communication schedule for the next round based on all active streams. The host transmits this schedule twice for increased reliability: in the last slot of the current round and in the first slot of the next round. Nodes use the first slot (*i.e.*, Glossy flood) in each round also to (re)synchronize with the host. As a result, all nodes share a common notion of time, which enables LWB's globally time-triggered operation. Experimental comparison against Muster, an traditional asynchronous many-to-many protocol based on multi-hop routing, demonstrates that LWB achieves a higher average data yield ($\geq$99.94 % versus 84–98 %) at a lower average radio duty cycle ($\leq$1.06 % versus 4–13 %) [39].

Completing the picture. Several protocols and services (see Sec. 4) adopt sequential Glossy floods from LWB, but use a different scheduler and modify the structure of rounds. For instance, in the Wireless Transparent Sensing Platform (WTSP) [128], a *sink* node schedules many-to-one communication toward itself, one-to-many dissemination from itself, and one-to-one "ping" messages between itself and any other node in the network. WTSP exploits the fact that one end of the communication is always the sink for end-to-end retransmissions based on implicit acknowledgments. Blink [154] provides real-time communication among arbitrary sets of senders and receivers. Stream requests have an additional *relative deadline* that specifies the maximum time between the generation of a packet and its reception. The Blink scheduler allocates slots in an earliest-deadline-first manner, using an admission-control scheme to ensure that all stream requests can be served. With this Blink guarantees that any *received* packet meets its deadline.

A few many-to-many protocols depart from sequential one-to-all flooding. For example, Byte-Cast [108] targets the many-to-many exchange of small data items, using an approach where senders adjust their transmit power *during* a synchronous transmission so a receiver can capture data items from different senders. Instead, Codecast [94] and Mixer [47] employ network coding to achieve a better scalability with the number of messages to be exchanged (see Sec. 3.3.2 for details). Although these protocols deliver every message to all nodes by default, packet filtering as in LWB can be used to only pass messages to the higher layers if the node is an intended receiver.

3.1.4 Many-to-one. Protocols in this class enable multiple source nodes in the network to transmit data to a sink node, which may provide connectivity to edge and cloud computing infrastructure.

Example → Crystal: Crystal [52] is a data collection protocol for aperiodic traffic. Nodes wake up at the beginning of an *epoch*, which consists of a short active phase and a long sleep phase. The active phase starts with a Glossy flood by the sink to synchronize the nodes. One or multiple pairs of slots follow. The first slot of every pair is similar to a contention slot in LWB: Multiple nodes with pending traffic may concurrently initiate a Glossy flood with a *different* packet. In the second slot, the sink acknowledges the reception of one of the packets due to the capture effect—or the reception of no packet—through another Glossy flood. Nodes whose packet is not acknowledged retransmit it in the first slot of the next pair. The sink enters the sleep phase if it does not receive any packet for a pre-defined number of consecutive slots. A distributed termination policy ensures that all other nodes also go to sleep after a certain number of slots with no network activity.

Crystal demonstrates that this functioning can translate into a highly energy-efficient runtime operation when combined with data prediction. The key idea of data prediction is that the sink estimates the physical quantity being sensed using a model. Thus, instead of periodically reporting its sensor readings, a node only sends a model update (*i.e.*, metadata) to the sink if the difference between the sensor reading and the model prediction exceeds a threshold. Previous studies have shown that this approach can significantly reduce the number of data transmissions [106, 110]. As aperiodic model updates should be received by the sink, an enhanced version of Crystal [53] adds channel hopping and noise detection to increase the protocol's resilience to interference.

Completing the picture. Another synchronous transmission based protocol that targets many-to-one communication is Choco [129], in which the sink schedules a Glossy flood for each packet including end-to-end retransmissions. The scheduler in Sleeping Beauty [112] also runs at the sink. It uses information about each node's one-hop neighbors to determine connected subsets of active nodes that transmit their packets, one after another, to the sink using Glossy-like floods, while all other nodes are put to sleep. FLEET [111] lets selected nodes act as cluster heads that first collect individual messages of a subset (*i.e.*, cluster) of nodes and then flood the aggregated data to the sink in a single packet. Packet-in-Packet (PiP) [88] adjusts the transmit power during a synchronous transmission to forward multiple smaller packets in one go towards the sink. The protocol proposed by Zimmerling et al. [152] directly builds upon LWB [39] and assumes that the sink and the host are the same physical device. The protocol also performs end-to-end retransmissions. Moreover, each node piggybacks a few quantities about its runtime operation on the packets. Feeding this information into analytical models, the sink can provide at runtime probabilistic reliability guarantees and predict the long-term energy consumption of each node.

3.1.5 All-to-all. Protocols in this class enable all nodes in a network to share data with each other, forming the basis for network-wide agreement or for computing a network-wide aggregate.

Example → Chaos: The first synchronous transmission based protocol specifically designed for this purpose is Chaos [71]. In Chaos, a designated node starts an all-to-all interaction by transmitting its message. When a node receives a message, it combines its own data with the received data

according to a *merge operator*, then transmits the combined data synchronously with other nodes. A merge operator can be, for example, a function that computes the maximum of a set of numbers or that concatenates a node's own and the received data. To avoid unnecessary transmissions, nodes are only allowed to transmit if they have new information to share. Whether there is new information to share is determined using *flags* inside each message, where each flag is mapped to a node in the network: A node sets a flag to 1 if it has (directly on indirectly) received the data of the corresponding node by computing the XOR of its own flags and the received flags. Once a node sees that all flags are set to 1 (*i.e.*, it has received data from all nodes), it blindly transmits the resulting message multiple times to allow other nodes to quickly reach completion. In this way, Chaos outperforms a collect-process-disseminate solution using traditional asynchronous protocols, CTP and Drip, by 17–22× in terms of radio duty cycle and 3–20× in terms of latency [71].

Completing the picture. The many-to-many protocols from above also support all-to-all communication. The main difference is that these protocols are meant for exchanging messages—they are pure communication protocols—whereas Chaos also performs in-network processing, allowing it to combine the data from all nodes into a single aggregate that is ultimately known to all nodes.

3.2 Traffic Pattern

We next classify synchronous transmission based communication protocols according to their traffic pattern. We thereby refer to the temporal pattern of transmitted packets *inside* the network as determined by the protocol logic, which may be different from the pattern with which packets are generated by the application. We distinguish protocols that transmit packets

- in a *time-triggered* fashion, including periodic and aperiodic traffic as governed, for example, by a schedule that is computed online or offline to determine when packets can be sent;
- in an *event-triggered* fashion, that is, packets can be sent instantaneously at any point in time with respect to the timescale of a packet's airtime (*i.e.*, initial delay of a few milliseconds).

3.2.1 Time-triggered. These protocols rely on a network-wide notion of physical or logical time to plan in advance when packet transmissions can occur. Typically, this information is encoded in a global communication schedule, which is distributed to all nodes to drive the network operation.

Example → LaneFlood: LaneFlood [16] uses a network-wide schedule to carry data traffic to and from nodes in an aperiodic fashion. Time in LaneFlood is divided into *sessions*, and each session contains several *rounds*. Communication occurs at the end of each round in allocated *communication slots*. A node designated as the initiator always sends a message during the first slot of each session. This message is used to synchronize the network and to let the initiator indicate whether or not it has data to transmit. If it has, the intended destination of the communication attempt replies, resulting in a handshake similar to CXFS [18]. If it the designated initiator has no data, other nodes with data use the second slot in the session to send their communication requests.

After the handshake, nodes in LaneFlood remain active only if they belong to the forwarder set, defined as in CXFS. However, instead of considering a fixed boundary b, LaneFlood lets nodes on a non-shortest path join the forwarder set according to a given probability. To tune this probability LaneFlood exposes a protocol parameter called the *slack*. This mechanism allows LaneFlood to include fewer nodes than CXFS in the forwarder set, saving energy without sacrificing reliability.

Completing the picture. Indeed, Table 1 shows most synchronous transmission based protocols exhibit a time-triggered traffic pattern. While some of these protocols focus on periodic traffic, such as Blink [154] and Sleeping Beauty [112], others are geared toward aperiodic traffic, such as WTSP [128] and Crystal [52, 53]. Nevertheless, they all plan well in advance when end-to-end transmissions can occur, possibly without specifying the concrete sender. The main reason is the

need to minimize idle listening when using active radios that draw significant power even when there is no ongoing packet transmission or reception.

3.2.2 Event-triggered. A few event-triggered protocols exist that do not require a persistent, global notion of time and enable nodes to transmit packets without any prior request or advance planning.

Example → Zippy: Zippy [123] provides on-demand one-to-all communication: individual packets are flooded upon an asynchronous network wake-up triggered by the sender. To this end, Zippy addresses several challenges associated with enabling synchronous transmissions using low-complexity on-off keying (OOK) transmitters and always-on ultra-low power OOK receivers serving as wake-up radios. When the network is idle, the transmitters of all nodes are off, but their wake-up receivers are still able to detect the presence of an incoming packet transmission while dissipating very little power (*i.e.*, on the order of a few microwatts). When a node detects an event, it simply starts sending its message with a specific preamble using its OOK transmitter. Neighboring nodes detect this preamble using their wake-up receivers, quickly turn on their transmitters, and then participate in propagating the message through the network using synchronous transmissions.

Zippy is suited to rapidly disseminate rare, unpredictable events. Time-triggered protocols would waste significant amounts of energy to capture these events with low latency. This is because the nodes would need to wake-up frequently to check whether there is an event to be disseminated. However, since events occur only very rarely, the energy spent for these wake-ups is wasted.

Completing the picture. The Event-based Low-power Wireless Bus (eLWB) [126] supports event-triggered traffic, based on a commodity node platform with a single active radio. When idle, eLWB schedules very short periodic rounds that contain only one *event contention* slot. When the need to communicate arises (*e.g.*, when an event is detected), one or multiple nodes send the same short, pre-defined packet in the event contention slot. Then, eLWB immediately schedules an *event round*, allowing all nodes to transmit their event and possibly request more bandwidth in future rounds.

BLITZ [124, 125] combines the wake-up radio technique from Zippy [123] with Glossy-based data dissemination. Specifically, the wake-up radio is only used to coordinate the start of a LWB-like communication round. Compared with eLWB, BLITZ spares the overhead of periodic communication rounds, as these occur only when at least one event of interest is detected. This approach greatly reduces the worst-case latency in communicating the occurrence of an event.

3.3 Type of Packets

Motivated by the physical-layer phenomena enabling synchronous transmissions and the associated requirements affecting protocol design (*e.g.*, in terms of timing), we distinguish protocols that

- ensure that always only *identical* data packets overlap in a synchronous transmission;
- allow for *different* data packets to overlap in a synchronous transmission.

As before, we refer to the *intent* of a communication protocol, not to what may accidentally happen at runtime. Furthermore, we restrict our attention to what happens during the communication of application-level *data*, as opposed to during the exchange of a protocol's control packets.

3.3.1 Identical. Table 1 shows that most of the protocols we survey intend to transmit identical data packets. One reason might be that the papers that made synchronous transmissions popular (*e.g.*, [31, 41]) focused on exploiting non-destructive effects when identical packets collide.

Example → PEASST: A notable example of a protocol that transmits only identical data packets is PEASST [61]. It enables point-to-point data transfers by exploiting low-power probing (LPP) to wake up the network before data communication. LPP uses a three-way handshake to establish a connection between senders and intended receivers. Receivers wake up periodically and send probe

messages to indicate their availability to receive data. A sender waiting for a specific receiver to wake up reacts to this probe by sending a reply that includes the address of the intended destination and forces nodes that receive it to keep their radios on. The intended receiver finally confirms the reception of the message from the sender and data exchange can start.

The handshake between receivers and senders allows the protocol to determine which subset of nodes must stay awake for a specific sender-receiver pair to communicate, similar to CXFS [18] and LaneFlood [16]. Using this mechanism, PEASST can, in principle, enable different sender-receiver pairs to communicate concurrently, whereas each data transfer is mapped onto floods of identical data packets. This differentiates PEASST from approaches like Mixer [90] and Chaos [71], which intentionally let different data packets overlap both temporally and spatially in the network.

Completing the picture. Glossy and most protocols that build on it fall into this category. For instance, although in LWB multiple nodes may initiate a Glossy flood in the same contention slot (see Sec. 3.1.3), the transmitted packets carry *control* information. By contrast, the first slot of each pair in Crystal is intended for the transmission of application-level data (see Sec. 3.1.4), and thus Crystal falls into the other category. Other noteworthy examples are protocols exploiting pipelined flooding, such as P^3 [26] and Pando [28, 29]. Using these protocols, different data packets co-exist *spatially* in the network, but those that *temporally* overlap at a receiver are always identical. Thus, these protocols aim to meet the tight timing requirements of constructive interference (see Sec. 2).

3.3.2 Different. A number of very recent protocols use synchronous transmissions of different data packets. These protocols are subject to the looser timing requirements of the capture effect, yet they must be cautious about the number of synchronous transmitters to meet the power requirements.

Example → Mixer: A protocol that lets different data packets occur in synchronous transmissions is Mixer [47, 90]. In the very beginning of a Mixer round, nodes transmit only the original messages. However, as soon as nodes receive more messages from their neighbors, they start transmit linear combinations of multiple original messages. To do so, nodes mix packets using random linear network coding (RLNC) [1, 49]. Since each node randomly, independently decides which packets to mix, nodes generally transmit different packets. A successful reception of these packets is still possible because of the capture effect, as the transmissions are sufficiently well synchronized and Mixer steers the number of synchronous transmitters based on short-lived information about a node's neighborhood. Nodes can decode the original messages from the received, mixed packets using the coding vectors embedded in the packet headers, which indicate what messages are coded together. In this way, Mixer approaches the order-optimal scaling with the number of messages to be exchanged in a many-to-many fashion across dynamic wireless mesh networks.

Completing the picture. Codecast [94] also provides many-to-many communication, but uses Fountain coding instead of RLNC, leading to a different scaling behavior. Nodes in Chaos [71] send uncoded messages in a slotted fashion like in Mixer or Codecast, yet combine data from different nodes according to a merge operator. Both ByteCast [108] and Packet-in-Packet (PiP) [88] use a packet concatenation scheme to compact small amounts of data from multiple neighboring nodes into a single packet.

3.4 Reliability Mechanisms

As described in Sec. 2, the high reliability of synchronous transmission based protocols cannot be explained by *sender diversity* alone. In addition, *receiver diversity* occurs whenever a transmission is overheard by more than one node, regardless of the number of senders. Further, all protocols we survey leverage one or multiple of the following mechanisms to achieve high reliability:

- local (*i.e.*, one-hop) or end-to-end packet *retransmissions*;

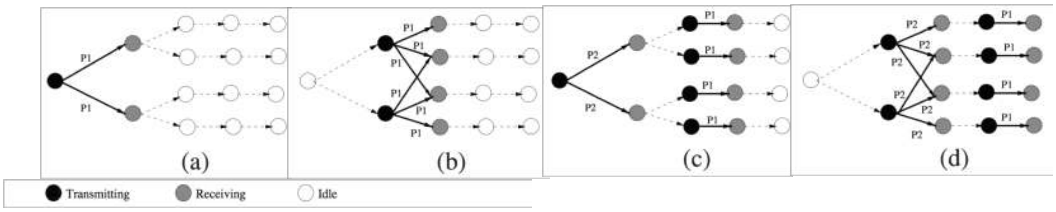


Fig. 5. Splash operation (taken from [27]). *Splash creates parallel pipelines, as indicated by packets P1 and P2 in the figure, over multiple paths covering all the nodes in the network. As basis for these parallel pipelines, Splash exploits the routing tree structure previously established by an existing one-to-all data collection protocol.*

- *channel hopping* to exploit frequency diversity;
- *network coding* for increased resilience to packet loss at the same communication costs.

3.4.1 Retransmissions. Despite very few exceptions, all communication protocols in Table 1 use packet retransmissions (see Table 1), albeit in vastly different flavors as discussed below. Retransmissions exploit *temporal diversity*: If a transmission failed, then chances are that some time later the retransmission may not suffer from the same effect that caused the previous one to fail.

Example → Probabilistic reliability guarantees: Zimmerling et al. [152] explore the analytical modeling of LWB to predict long-term energy consumption and to provide probabilistic reliability guarantees. This is feasible because of two main reasons. First, unlike link-based protocols, LWB's protocol logic is independent of the time-varying network state (link qualities, neighbor tables, etc.), so the network dynamics play no role in the modeling. Second, the authors show empirically that, unlike link-based transmissions, packet losses and receptions in Glossy can be faithfully characterized by a series of independent and identically distributed (i.i.d.) Bernoulli trials. Later confirmed by a theoretical study [63], this property enables simple, yet accurate analytical models.

To give probabilistic reliability guarantees, Zimmerling et al. extend the original LWB scheduler in a setting where the host is also the sink. At the end of each round, the scheduler allocates missing packets another slot in the next round, up to k_{max} times. This is taken at the sources as an implicit acknowledgment that the sink has not received the corresponding packet, requiring an end-to-end retransmission. As the Bernoulli property holds, it is easy to derive an appropriate k_{max} value for all streams given a desired end-to-end reliability, as determined by application requirements [152].

Completing the picture. A number of protocols use the same flavor of end-to-end retransmissions, including Crystal [52, 53], Choco [129], and WTSP [128]. These protocols and many others based on synchronous transmissions also benefit from local (*i.e.*, one-hop) retransmissions; for example, in Glossy [41], RedFixHop [37], or Zippy [123] every node in the network *opportunistically* retransmits a packet up to a certain number of times. Local retransmissions may also be triggered by an explicit *request* as used in Codecast [94], Mixer [47, 90], or Splash [27] to collect a few packets that some nodes still miss toward the end of a many-to-many or one-to-all communication phase.

3.4.2 Channel hopping. This is a common technique to address sources of unreliability, such as interference from co-located networks and background noise [12], through *frequency diversity*: While one channel may be affected by such phenomena, likely other channels are less affected [105].

Example → Splash: The Splash protocol [27] uses pipelined Glossy floods for fast and reliable dissemination of large data objects (*e.g.*, a program image). The source divides the data object into multiple smaller packets. Nodes at the same hop distance from the source form a *layer*, and every other layer simultaneously transmits a different packet, as shown in Fig. 5. This way, Splash creates parallel pipelines over multiple paths that cover all the nodes in the multi-hop network.

Inspired by the pipelined transmission scheme from [105], membership of a node to a layer is decided based on an existing routing tree created by a traditional link-based data collection protocol (*e.g.*, [43]). Splash assumes that such collection protocol runs between the dissemination phases, and the root of the tree is taken as the dissemination source. A pre-determined channel assignment that maps a layer to a channel is distributed prior to each dissemination round. Consecutive layers in the pipeline are assigned the same channel, whereas channel changes across the pipeline employ no adjacent channels. This makes Splash reduce internal interference, support deeper pipelines, and work around interference that extends across several hops. In addition, Splash also incorporates a scheme to reduce the number of transmitters in areas with a high node density, opportunistic overhearing of transmitted packets in the same layer, and XOR coding of packets (see Sec. 3.4.3).

Completing the picture. Several synchronous transmission based data dissemination and bulk-transfer protocols, which need to transmit multiple packets from the same source, use a similar channel-hopping approach in combination with pipelined Glossy-like flooding, such as Ripple [146], Pando [28, 29], and P^3 [26]. BlueFlood [4] performs channel hopping while flooding a single packet: Every node switches to a new channel in each time slot *within* a flood according to a network-wide hopping sequence, which covers all 40 channels available in Bluetooth 5. By contrast, other protocols like Crystal [53] and Harmony [87] change channel only *between* consecutive floods.

3.4.3 Network coding. Rather than forwarding messages as independent units through the network, communication protocols using network coding deal with packets that are combinations of multiple messages [1]. To this end, senders, receivers, and relay nodes encode and decode packets, depending on the network-coding technique used. In general, network coding helps achieve higher throughput or higher reliability (or a middle ground) at the same communication costs compared to the non-coding case. It has been found that network coding is particularly suitable for wireless and sensor networks, for example, due to the broadcast and lossy nature of the wireless medium [42, 64].

Example → Pando: Similar to Splash [27], Pando [28, 29] disseminates large data objects using pipelined Glossy floods. Instead of distributing the original messages that constitute the data object, the source injects encoded packets built by encoding a specific number of randomly selected original packets. All other nodes relay and incrementally decode these packets as they arrive, using a variant of the Gaussian elimination algorithm. Nodes recover the data object when they receive as many linearly-independent linear combinations as there are original messages. Afterward, Pando uses a silence-based feedback scheme to inform the source that the dissemination process can stop. Unlike Splash, where nodes use different channels but stay on a specific one for an entire dissemination round, nodes in Pando switch channel on a per-packet basis to better adapt to channel variations.

Completing the picture. Only a few other communication protocols exist in the literature that combine synchronous transmissions with different forms of network coding. While Pando [28, 29] leverages Fountain coding, Splash [27] uses XOR coding and Ripple [146] uses a Reed-Solomon code for fast and reliable one-to-all data dissemination. Codecast [94] and Mixer [47, 90] target fast, reliable many-to-many communication, using Fountain coding and RLNC, respectively. Both protocols exploit feedback from neighbors, enabling a sender to build encoded packets that are likely useful for its neighbors, that is, help toward a fast completion of the many-to-many exchange.

3.5 Network State

Whether the logic of a protocol relies on information about the state of the network is a key factor determining the protocol's ability to adapt to changes resulting from moving or failing nodes, time-varying external interference, etc. Traditional link-based protocols maintain network state to *avoid* packet collisions, for example, by scheduling transmissions between individual sender-receiver

pairs based on information about a node's one-hop neighbors and the quality of the links between the node and its neighbors. Instead, synchronous transmissions *embrace* packet collisions, enabling the design of communication protocols that maintain no network state at all. Nevertheless, the design space of existing protocols is diverse, motivating us to distinguish protocols whose logic

- is independent of any information about the network state, referred to as *none*;
- depends on *static* network state information that is not updated by the protocol;
- depends on *dynamic* network state information that is updated by the protocol.

We note that while a protocol's *logic* may be independent of the network state, its *performance* always depends on the network state. For example, the time and energy needed to transmit a packet from one node to another inevitably depends on the hop distance between the two nodes.

3.5.1 None. Protocols in this class are oblivious of the network state, allowing them to seamlessly operate in the presence of significant network and environment dynamics.

Example → BlueFlood: The work on BlueFlood [4] explores through controlled experiments the applicability of synchronous transmissions to the physical layer of Bluetooth 5 and the impact of several factors, from time offsets to packet contents. The authors then build on these insights and propose a Glossy-like one-to-all protocol that maintains no network state. Different from Glossy, every node in BlueFlood performs a certain number of back-to-back synchronous transmissions after receiving a packet, which increases efficiency compared with Glossy, and switches channel after each transmission and reception for increased reliability, as discussed in Sec. 3.4.2. The nodes must keep track of round and slot numbers to follow the global channel hopping sequence, yet this kind of state only depends on the progression of time and not on the time-varying network state.

Completing the picture. Looking at Table 1, we find that many of the communication protocols we survey maintain no network state. All of them adopt some form of synchronous transmission based one-to-all flooding, either individually to transmit a single packet [37, 41, 123] or in a series to transmit multiple packets [39, 124–126, 128, 129, 152, 154]. The only exception is Chaos [71], where nodes transmit different packets in a slotted fashion for all-to-all comunication.

3.5.2 Static. A few protocols operate on network state information that is assumed to be static for extended periods of time and not updated by the protocol itself. One of the goals of these protocols is to limit the number of nodes involved in flooding a given packet to improve efficiency.

Example → SCIF: The Spine Constructive Interference-based Flooding (SCIF) protocol [136, 137] applies graph algorithms to identify a backbone to funnel packets through. The authors observe analytically that involving all nodes in the flooding creates reliability problems as the number of synchronous senders increases. To address this issue, SCIF determines a network *spine*, a subset of the nodes showing specific topological characteristics that act as a backbone for the flooding. Unlike in Glossy, nodes outside of the spine remain silent after receiving a packet. This results in fewer synchronous senders, improving reliability according to the author's analytical results.

To determine the spine, SCIF assumes that node locations are *known a priori* and *do not change* over time. With this information, and assuming a unit-disk graph model with known communication range, the network topology is mapped to a graph structure. The spine is determined by dividing the resulting graph in smaller cells with a known ratio to the communication range, and by enforcing cells to form a connected sub-graph through at most one link between adjacent cells. Any change in the physical network topology, especially due to node mobility, would break SCIF's functioning.

Completing the picture. Splash [27] and Pando [28, 29] do tolerate network state changes, but only *between* data dissemination phases, which can take a few tens of seconds or even longer for a program image. Both protocols rely on a tree topology, maintained by *another* collection protocol,

which must remain intact throughout the protocol execution, as described in Sec. 3.4.2. None of the two protocols adapts the tree topology to network state changes during the data dissemination.

3.5.3 Dynamic. Protocols exist that seek to improve performance by leveraging network state, which they update dynamically to ensure that it is consistent with the physical network topology.

Example → P^3: The Practical Packet Pipeline (P^3) protocol [26] is similar to Splash, yet optimized for one-to-one bulk transfer. Noting the same reliability problems in dense scenarios reported in earlier works [27, 136], P^3 exploits a sub-group of nodes that form a set of node-distinct paths of a given hop length, connecting source and destination. Based on knowledge of the network topology, obtained at a base station during a discovery phase, P^3 uses depth-first search to find the largest set of available node-distinct paths of a given hop length. If no such path exists, the procedure repeats by increasing the desired hop length until a minimum number of node-distinct paths is found. All nodes that do not lie on such paths do not participate in the data transfer. The selection of node-distinct paths is then communicated to the nodes before the bulk transfer commences. The operation of P^3 is therefore a function of how accurate is the topology information used to identify the node-distinct paths. For example, if a node moves between the time topology information is acquired and the bulk transfer commences, the node-distinct paths are no longer guaranteed to feature the same characteristics. As in Splash, transmissions are pipelined over multiple channels to achieve high throughput. In addition, the receiver uses negative acknowledgements (NACKs) to notify the sender of missing packets, thus ensuring reliable transfer of bulk data.

Completing the picture. Other protocols including CXFS [18], LaneFlood [16], Whisper [15], PiP [88], and Spleeping Beauty [112] also exploit hop counts or link qualities dynamically collected during a discovery phase. The ability to still tolerate node mobility and other dynamics relies on the time span over which such network state must remain consistent. For example, nodes in Mixer [47, 90] keep a short history of what happened in the last couple of slots during a many-to-many communication phase to make informed protocol decisions, yet the time span of the history (tens of milliseconds) is so short that nodes moving at a speed of 60 km/h do not affect the protocol's performance [47].

4 SYNCHRONOUS TRANSMISSIONS IN NETWORK SERVICES

The use of synchronous transmissions extends beyond the design and implementation of communication protocols. The literature includes a range of network services employing synchronous transmissions for a variety of purposes. We broadly categorize these works in the following classes, and provide a representative from the existing literature to make the discussion concrete. Table 2 comprehensively lists the works we discuss here, along with their classification.

4.1 Consensus

Many communication protocols based on synchronous transmissions implement a form of synchronous behavior to coordinate operations and employ network-wide flooding to distribute application information. These features incidentally bring the system one step closer to the synchronous broadcast model that traditional distributed systems literature builds upon [11]. Functionality such as virtual synchrony [40], two- and three-phase commit [3, 117], as well as Paxos agreement [101] thus become possible. Notably, prior to the emergence of synchronous transmissions, literature argued that achieving these functionality in low-power wireless networks was exceedingly difficult [121].

Example → A^2: The Agreement-in-the-Air (A^2) protocol [3] provides a means to reach network-wide consensus, for example, as implemented using two- and three-phase commit, based on synchronous transmissions. It supports applications with low-latency and reliable data delivery

Table 2. Overview of surveyed network services using synchronous transmissions, classified according to the kind of service they provide. Approaches marked in bold are those described as the representative example of their own service class.

System	Service class	Discussion
Virtus [40]	Consensus	Sec. 4.1
A^2 [3]	**Consensus**	
Paxos Made Wireless [101]	Consensus	
XPC [117]	Consensus	
DRP [57]	Time-bounded coordination	Sec. 4.2
TTW [55, 56]	**Time-bounded coordination**	
DMH [139]	Time synchronization	Sec. 4.3
Reverse flooding [130]	**Time synchronization**	
TATS [83]	Time synchronization	
PulseSync [76, 77]	Time synchronization	
SurePoint [65]	**Localization**	Sec. 4.4
Chorus [21, 22]	Localization	
SnapLoc [45]	Localization	
pTunes [153]	**Network management**	Sec. 4.5
Atomic-SDN [7]	Network management	
A-MAC [31–33]	**Medium access**	Sec. 4.6
NDI-MAC [85]	Medium access	
PiP [151]	Medium access	
Aligner [84]	Medium access	
TriggerCast [135]	Per-hop reliability	Sec. 4.7
CIRF [144]	**Per-hop reliability**	
Disco [138]	Per-hop reliability	
Backcast [33]	Collision resolution	Sec. 4.8
Strawman [99]	**Collision resolution**	
Stairs [62]	Collision resolution	
POC/POID [143]	Collision resolution	

requirements. A^2 builds on a synchronous transmission kernel called Synchrotron, which extends the Chaos [71] protocol with high-precision synchronization, time-slotted operation, a network-wide scheduler, frequency hopping, multiple parallel channels, and security features.

On top of Synchrotron, A^2 builds a voting protocol used to implement two- and three-phase commit, functionality for consistent group membership, and reliable primitives for nodes to join and leave the network. Applications use A^2 to reliably agree on, for example, cryptographic keys, channel-hopping sequences, or actuator commands, even in the presence of node or link failures.

The authors demonstrate that A^2 incurs in a 475 ms latency to complete a two- phase commit over 180 nodes in the presence of unreliable links. This comes at an energy cost that does not exceed a 0.5% duty cycle for 1-minute intervals. When adding controlled node failures, they also show that two-phase commit as implemented by A^2 ensures transaction consistency, while three-phase commit provides liveness at the cost of possible inconsistency in specific failure scenarios.

4.2 Time-bounded Coordination

Existing literature also aims at capitalizing on the synchronous behavior at the root of many communication protocols using synchronous transmissions by extending the corresponding semantics to application-level functionality [55–57]. Especially whenever real-time requirements are at stake, the ability to comprehensively cater for timing aspects across application and network, and in

a distributed fashion, allows systems to operate on a single clock reference. This enables better resource utilization and more accurate scheduling.

Example → TTW: Jacob et al. [55, 56] design a system that operates across the stack, from application down to individual packet transmissions, in an attempt to bring the operation of a low-power wireless network as close as possible to that of a wired field bus. To this end, three issues are to be solved: *i)* dedicated scheduling techniques need to be devised to match the round-based execution used in low-power wireless to reduce energy consumption, *ii)* pre-computed schedules must be determined as computing schedules entirely at run-time may be prohibitive because of resource and time constraints, and *iii)* adaptive mechanisms are to be created to cope with wireless dynamics and changing requirements.

Using Integer Linear Programming (ILP) techniques, Jacob et al. formally specify and solve the scheduling problem across distributed application tasks, messages flowing in the network, and communication rounds. The corresponding solution guarantees application timings to be within stated application requirements, minimizes end-to-end latency across the data flows between application tasks, and ensures safety in terms of conflict-free communication, also in the presence of packet losses. Time-Triggered-Wireless (TTW) is a system design that incorporates the solution to the scheduling problem, therefore aiming at fulfilling staple requirements of distributed applications running on low-power wireless networks, such as reliability, timing predictability, reduced application-level end-to-end latency, energy efficiency, and runtime adaptability. The formulation and solution to the scheduling problem, as well as the TTW design, build on top of LWB [39], as they inherit its round-based operation and the use of a host node for coordination.

The evaluation is based on simulations obtained with time and energy models and aim at quantitatively measuring the minimal latency attainable across application tasks and the energy savings obtained from round-based communication. The results indicate, for example, a minimum latency of 50 ms in a 4-hop network using 5 communication slots. The authors expect the simulation results to be on par with the actual performance of a concrete implementation, based on the quantitative observations obtained from synthetic models of LWB [152].

TTW has been instrumental for the first work that demonstrates fast feedback control and coordination with closed-loop stability guarantees over real multi-hop low-power wireless networks [89], provding global schedules for application and communication tasks with negligible end-to-end jitter and minimum end-to-end latency also for multi-mode CPS [8].

4.3 Time Synchronization

Leveraging the synchronization already required among synchronous senders to reap the benefits described in Sec. 2, many communication protocols based on synchronous transmissions already provide a form of time synchronization [27, 41]. Works exist that take this one step further, and focus exclusively on employing synchronous transmissions to achieve accurate system-wide time synchronization [76, 77, 83, 130, 139].

Example → Reverse flooding: Terraneo et al. [130] build on the observation that an accurate implementation of synchronous transmissions for network-wide flooding must also remove time inaccuracies at MAC layer. This is necessary to retain the temporal alignment of synchronous transmitters, as discussed in Sec. 3.1. As a result, the challenge of time synchronization moves from accurate local timestamping at MAC layer, as seen in previous literature [36], to the estimation of network propagation delays, which were previously considered a secondary issue.

To tackle this issue, Terraneo et al. design a compensation technique to account for such delays in a setting with one clock reference. The core of their solution lies in identifying the predecessor set of every node in the network during flooding, and then performing per-hop propagation delay

measurements similar to round-trip estimations. The resulting measurements are employed to infer a cumulated delay distribution used to estimate the propagation delays at every hop.

Experimental evidence obtained using CC2520 transceivers and Cortex M3 MCUs in a variety of configurations indicates that their network propagation delay technique allows to cancel up to 95% of the error induced by propagation delays for a generic clock synchronization scheme. This allows allows Glossy-like time synchronization schemes to achieve sub-microsecond clock synchronization.

4.4 Localization

Works exist that employ synchronous transmissions to assist the localization in space [65] or to perform key steps of the localization process [21, 22, 45]. In these cases, synchronous transmissions become a stepping stone to the development of protocols coordinating the ranging operation or a foundation to obtain better accuracy or higher frequency of location updates. We briefly describe next an example of the former kind of approach.

Example → SurePoint: Kempke et al. [65] present a system providing decimeter-accurate localization based on the time-of-arrival information provided by ultra-wide-band (UWB) radios, improved by frequency and polarization diversity through multiple communication channels and multiple antennas.

Compared to earlier works in RF-based localization [92], SurePoint takes a step further by supporting the concurrent localization of multiple tags. As localization of a single UWB tag requires exclusive access to the radio spectrum, the authors borrow the idea of time-triggered mutually-exclusive operation of LWB [39], described in Sec. 3.1. In essence, SurePoint replicates LWB's scheduling technique over UWB radios to determine what single tag is being localized at a given point in time, similar to the way LWB grants exclusive access to a single source at a time. Procedures for joining the network, reaching and maintaining steady-state operation, and leaving the network are also provided, similar to LWB.

The authors demonstrate that SurePoint can handle up to 10 tags with no impact on localization accuracy, reaching steady-state operation in only 10 seconds even when all tags enter the system at the same time. Similar to LWB, however, the authors observe that scheduling tag localization in a exclusive-access manner scales linearly with the number of tags, which may eventually cause scalability problems. This is an effect of the bus-like scheduling that the use of synchronous transmissions often leads to, as observed in multiple occasions [40, 152].

4.5 Network Management

Dedicated protocols using synchronous transmissions are possibly employed as the management plane for other network functionality, for example, to distribute new system parameters or to collect run-time statistics [7, 153]. The key advantage in this case is the ability to decouple the operation of the main network functionality from the execution of the management protocol using synchronous transmissions. We present a representative example next.

Example → pTunes: Zimmerling et al. [153] design a framework enabling runtime adaptation of traditional MAC protocols that use link-based transmissions. Based on application requirements expressed as bounds on network lifetime, end-to-end latency, and end-to-end reliability, pTunes automatically determines optimized parameter values to adapt to link, topology, and traffic dynamics. At a central basestation, optimized parameters are obtained by solving a multi-objective optimization problem based on a model of the MAC protocol operation. The modeling approach separates protocol-dependent from protocol-independent aspects, thus facilitating the use pTunes with different MAC protocols.

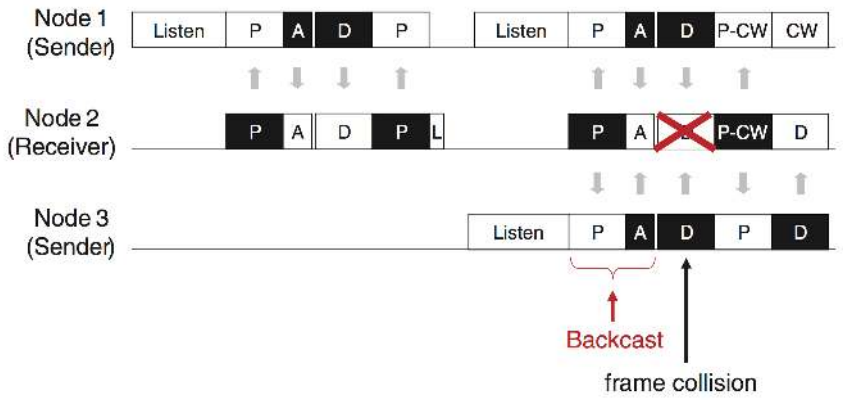


Fig. 6. Fundamental operation in A-MAC (taken from [32]). *The packet transmission on the left only includes a single sender. The probe packet **P** is correctly received and generates an auto-acknowledgement **A**. The data frame **D** is also correctly delivered, and a further probe packet **P** acknowledges the reception. The receiver listens for a short while before turning the radio off, labeled **L**. In the packet transmission on the right, two senders collide. The synchronous transmission of auto-acknowledgements allows the receiver to correctly decode the answer to the probe packet. A new contention window **CW** allows both senders to eventually transmit their packet.*

Zimmerling et al. design a custom system support to manage the runtime operation of pTunes. This is conceived to fulfil requirements involving timely and consistent collection of network state information from all nodes, to ensure the optimization process at the base station operates with fresh information, as well as reliable distribution of updated MAC protocol parameters back to all nodes, to guarantee that a MAC protocol coherently operates across the entire system.

To this end, they integrate single Glossy floods into the operation of pTunes. Based on Glossy's time synchronization service, collection of network state information and possible updates to MAC parameters are scheduled simultaneously at all nodes. Due to the fast execution of individual Glossy floods, this approach provides minimal disruption to the execution of the main MAC protocol and to the upper-level application. This is quantitatively demonstrated when using pTunes with X-MAC [17] and LPP [98] in periodic data collection scenarios.

4.6 Medium Access

A form of synchronous transmissions is sometimes employed at the MAC level to achieve efficient coordination within a single hop [31, 32, 84, 85, 151]. In this setting, synchronous transmissions allow one to implement quick rendezvous operations in an energy-efficient way.

Example → A-MAC: Dutta et al. [31, 32] present a receiver-initiated MAC protocol that, unlike previous MAC protocols of the same kind, employs a form of synchronous transmissions to coordinate the probing receiver with the impeding sender(s). The basic A-MAC operation, shown in Fig. 6, requires a potential sender to first listen for a probe packet from the intended receiver, then acknowledge the frame using the 802.15.4 hardware automatic acknowledgments, then pause for a short random delay, and finally transmit the data packet if the channel is clear.

The key lies in the use of automatic hardware acknowledgements. The 802.15.4 standard precisely dictates the content and format of such acknowledgement, as well as the time that it is generated after an application packet with the hardware acknowledgment bit set is received. Such a time is enforced in hardware by any 802.15.4-compliant radio chip. As a result, all nodes within the neighborhood of their intended receiver synchronously generate the acknowledgment. Even if

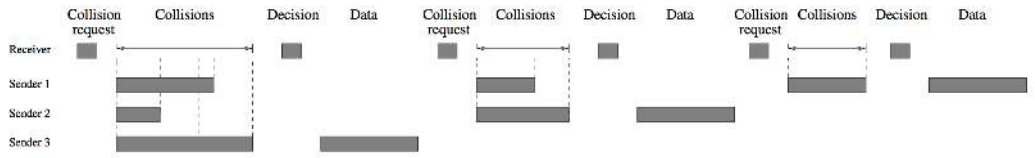


Fig. 7. Strawman operation (taken from [99]). *Upon detecting a collision, a receiver triggers synchronous transmissions of* Collision *packets from potential senders. Without concretely decoding the packet, a receiver uses the length of such packets to resolve contention.*

multiple such acknowledgments are generated, they are all identical and aligned in time; therefore, they interfere non-destructively and the receiver can decode its content with high probability.

This mechanism, called *backcast* [33] and shown in action on the right in Fig. 6, is crucial in A-MAC, as it is the basis for the protocol to decide whether a receiver needs to stay awake as at least one impeding transmission exists, or the radio can be turned off. Compared to how the same decision is taken in previous receiver-initiated MAC protocols, the use of synchronous transmissions in backcast results in quicker, more energy efficient, and more accurate operation. The latter holds particularly in the case of multiple senders, which would otherwise generate temporal misaligned responses to the probe packet that would be impossible to decode, preventing the receiver from understanding that a transmission is about to arrive.

4.7 Per-hop Reliability

Synchronous transmissions are also employed as a drop-in replacement for link-based transmissions in the context of protocol designs that would equally employ the latter [135, 138, 144]. Here, single-hop protocol interactions are adapted to make them compatible with synchronous transmissions without altering the traditional multi-hop protocol operation or its API.

Example → Cirf: Yu et al. [144] design a network-wide flooding protocol that employs a modified version of the RI-MAC protocol [122]. In an low-power wireless network using RI-MAC, it is likely the case that multiple nodes wait for the same neighbor to wake up when flooding. In the original RI-MAC, contending transmitters cause collisions and a dedicated backoff mechanism is put in place to remedy this issue. This increases packet retransmissions and latency, and is thus detrimental to energy consumption; yet, the authors observe that when these issues manifest because of network-wide flooding, the colliding packets likely carry at least the same payload.

To improve performance in this specific setting, the authors take advantage of the implicit time synchronization provided by RI-MAC's beacon packet within the one-hop neighborhood, and temporally align transmissions from multiple neighbors to leverage constructive interference, in addition to capture effect, for at least part of a packet, that is, the application payload. A node then sleeps only whenever all its neighbors received the current packet. This condition cannot be detected by only looking at application payloads; therefore, Cirf employs an ad-hoc packet header designed to retain the same structure across different neighbors, and thus also enjoying the benefits of constructive interference in addition to capture effect.

4.8 Collision Resolution

The literature includes works that use synchronous transmissions to realize efficient collision resolution mechanisms [33, 62, 99, 143]. In this case, synchronous transmissions are not employed to transmit actual application data, but to implement a signaling protocol invisible to upper layers.

Example → Strawman: Österlind et al. [99] present a collision resolution mechanism for receiver-initiated MAC protocols. Fig. 7 illustrates its operation.

Upon detecting a data collision, the receiver sends a Collision request packet. The senders interpret this packet as the beginning of a Strawman round, and contend for the channel by *synchronously* transmitting a Collision packet of *random* length. The receiver does *not* attempt to decode the packet; it only needs to estimate the length of the longest Collision packet by sampling the received signal strength. The receiver then broadcasts a Decision packet containing the longest measured length, implicitly informing the corresponding transmitter that it is now granted access to the channel.

While the selected transmitter transfers the Data packet, the other contenders remain silent, as they also recognized not to be given channel access based on the information in the Decision packet. The subsequent Collision request broadcast by the receiver both initiates a new Strawman round and acknowledges the previous Data packet. In the case that several contenders have chosen the same random length and their Data packets have collided, the receiver sends another Collision request packet to initiate a new Strawman round. This process repeats until all contenders have successfully sent their Data packets.

In Strawman, it is therefore merely the temporal alignment of Collision packets that ultimately allows the synchronously transmitted packet to carry some information, without the receiver concretely decoding the packets. Similar to A-MAC, the impact on the system's performance materializes as quicker and more energy-efficient operation compared to alternative approaches for contention resolution [59].

5 RESEARCH AGENDA

We make here a brief account of those, eliciting open problems and standing issues.

5.1 Network Stack and Operation

The design of a network stack based on synchronous transmissions is largely yet to be explored. The protocols and services in this article represent individual prototypes, but research to consolidate and integrate those efforts into a modular, full-fledged stack that exploits the characteristics of synchronous transmissions with a well-defined API is limited. There is likely no one-size-fits-all solution. However, to best serve emerging CPS and IoT applications, research should aim for a dependable, secure, and adaptive stack that supports dynamic networks and provides provable guarantees on performance and functional properties. Such stack currently does not exist; however, it is a challenging, yet realistic goal given the recent achievements in the field.

A notable effort in this direction is Baloo, a framework for the design of low-power wireless stacks based on synchronous transmissions [54]. At the core of Baloo is a middleware layer that allows one to compose multiple synchronous transmissions primitives, while hiding low-level interrupt, timer, and radio control from protocol designers. An additional question, however, arises:

> *What are the trade-offs between performance, functionality, and the number of synchronous transmissions primitives underneath?*

Exploration in this area may be pushed as far as providing networking, localization, and time synchronization using a single physical layer and synchronous transmissions primitive, such as one-to-all flooding over ultra-wideband radios [65, 133]. If multiple primitives are used, the increase in complexity may not necessarily justify the additional flexibility or performance improvements.

Most multi-hop protocols using synchronous transmissions also operate in a slotted fashion. The network-wide slot grid emerges as the transmissions of neighboring devices must be aligned in time. Protocols that rely on capture effect typically use explicit software routines to establish and

maintain the slot grid [47, 94], while those exploiting constructive interference exclusively rely on hardware interrupts [28, 41].

The network-wide slot grid serves as a global, shared timeline for scheduling the distributed protocol operation: In every slot, each node transmits, listens/receives, or turns off the radio. Despite the key role, mechanisms to establish and maintain the slot grid are often best effort and poorly understood. Although the nominal slot length is known to each node, the nodes must keep the slot grid in a fully distributed fashion, that is, with no central reference clock, despite varying clock offsets across nodes. This setting is different from most time synchronization protocols, where all nodes synchronize with a specified reference node. Further issues are thus worth exploring:

> *Is it possible to design and formally analyze a mechanism that guarantees all nodes stick to the slot grid within a certain tolerance range, despite network dynamics and as the number of nodes, number of slots, or topological network features vary?*

The problem resonates with recent work on self-organizing pulse-coupled oscillator synchronization [66] and synchronization in networks of phase-locked loops [102]. Therefore, we ask:

> *Are practical mechanisms based on those theories feasible, and can such an approach enable formal guarantees on convergence and synchronization accuracy?*

A carefully-designed slotting mechanism is a prerequisite for building a dependable and efficient network stack, and may elicit important bounds on, for example, network diameter and slot length.

5.2 Networking Abstractions

Wireless communications are, by their very nature, broadcast: The information carried by the radio waves may be received by all devices within communication range. However, the vast majority of today's networks, from (low-power) wireless local area networks to the Internet, is based on point-to-point interactions. Multicast and broadcast communication are only used for specific tasks, such as device discovery and data replication.

By contrast, synchronous transmissions break away from the point-to-point abstraction. Several solutions in Sec. 3 adopt a form of broadcast-only communication model: every message may be received by all other nodes in the (multi-hop) network. As discussed in Sec. 4.1, the combination of this and network-wide time synchronization narrows the gap between low-power wireless systems and traditional distributed ones, where a large body of exiting literature applies [11]. This enabled network-wide agreement [3] and consensus [101] as well as hard real-time [154], virtual-synchrony [40], and closed-loop stability guarantees [89]. These are desirable functionality in the target application, which appeared out of reach in low-power wireless given their dynamic nature and resource constraints [120]. This poses a key question:

> *Is the broadcast model a better abstraction for future low-power wireless networking using synchronous transmissions, or what other abstractions may reap the most benefits from synchronous transmissions?*

We argue that future research should reconsider this aspect in light of broadcast-only communication and network-wide synchronization. For example, Tschudin [131] points out the tight relationship between the broadcast model and replicated append-only logs: a data structure where every node keeps a full log of all packets ever sent by a given source. In an analogy with interpreting communication between two processes over a pipe as appending to a FIFO buffer, replicated append-only logs may allow to "... replace networking with arbitrary data packets by networking with coherent data structures" [131]. Barring memory limitations on the target platform, implementing replicated append-only logs appears feasible using synchronous transmissions, by relying on the property that each node may receive every message. This may pave the way to the principled

design and efficient implementation of a fully decentralized network stack that does not require central entities and is fault-tolerant under non-Byzantine conditions, including message losses, network partitions, and node failures.

Further examples are distributed shared memory [24] and the synchronous semantics underlying languages such as Lustre and Giotto [9], which may raise the level of abstraction for developers and allow them to reason about timing properties. The networking and system-level mechanisms required to efficiently implement those semantics should inform the design of the network stack and communication protocols. Putting those mechanisms in place would equate to providing access to a large body of prior work in distributed, embedded, and real-time systems that hitherto remained applicable only to conventional networks, enabling a better understanding and even formal verification of low-power wireless systems. Existing work shows that synchronous transmissions simplify accurate protocol modeling [152], which is often needed for verification.

On the other hand, broadcast-only communication and provisioning for the abstractions above may not be suitable for all applications, and scalability in terms of number of devices and traffic load may be a concern. The other side of the coin prompts us to ask:

> *What are the application facets that make would such a decentralized approach appropriate, given the achievable consistency model, performance, and timing guarantees may be different compared with a centralized approach?*

A better understanding of the conceptual and practical trade-offs between broadcast-only and point-to-point interactions, along with their breakeven point in real networks, if any, may be informed by recent theoretical results [5, 46].

5.3 Implications of Physical-layer Features on Protocol Design

The works in this article indicate some of the potential of using synchronous transmissions instead of (or alongside with) link-based transmissions to design protocols with unprecedented functionality, performance, and efficiency. As the community broadens its understanding of lower-level questions, such as why and under what conditions synchronous transmissions work for an increasing number of low-power wireless physical layers and receiver architectures [6], the corresponding insights may have profound implications on the design of higher-level communication protocols and network services. The problem is multi-faceted.

Physical-layer phenomena. It is important to examine how low-power wireless receivers benefit from constructive interference versus different phenomena, such as capture effect. Orthogonal to this, the effects of spatio-temporal diversity also warrant further investigations.

Based on current literature, we know that as the transmit bitrate of the physical layer increases, it becomes more and more difficult to meet the timing condition of constructive interference. For example, using Wi-Fi as physical layer, the incoming signals at a receiver must be aligned to within a few nanoseconds, that is, a fraction of the symbol length. To meet such stringent timing constraints, special hardware and accurate network state information are a necessity, for example, to compensate for different propagation delays [104]. A key question therefore arises:

> *Despite the added complexity, is it possible to employ constructive interference in networks with mobile entities or in deep networks with a diameter of more than a few hops?*

Even for low-rate physical layers, such as IEEE 802.15.4, it is unclear whether and to what extent the timing conditions of constructive interference are met beyond the second hop, for example, during a Glossy flood. Sophisticated experimental setups are needed to investigate this question as

currently available testbeds such as FlockLab [82] do not typically provide the required synchronization accuracy. We conjecture that ideas from time-of-flight-aware time synchronization [83] may help increase the chances that nodes benefit from constructive interference.

Nonetheless, several arguments suggest to design protocols that exclusively build upon the capture effect. Compared with constructive interference, the capture effect is more general and has significantly looser timing conditions, corresponding to the length of the packet preamble. This simplifies protocol design and implementation, especially in light of emerging hardware platforms with higher clock rates, and also leaves headroom for incorporating compute-intensive security features, such as packet encryption and authentication [3], network coding operations [47, 94], and other in-network processing functions [71, 142]. Although further studies on capture thresholds and windows are needed [6], it also becomes easier to port a protocol to other platforms, to support physical layers with a higher transmit bitrate, and to run a protocol in a heterogeneous network consisting of nodes with different clock rates. About capture effect, however, we ask:

> *If capture is the dominating or even the only effect that enables packet reception beyond a certain hop distance, what does this mean for protocol scalability and reliability, particularly in dense networks?*

On the contrary, existing multi-hop protocols that are exclusively based on the capture effect typically use some form of probabilistic decision making, for example, whether a node should transmit or stay silent, to increase the chances that the capture effect occurs [47, 71]. This makes their timing behavior non-deterministic and thus harder to predict accurately compared with other solutions that also build upon constructive interference [152, 154]. Further research into understanding and (ideally) bounding the running time of these protocols is needed, for example, to assess their applicability in hard real-time applications.

The message-in-message effect essentially plays like the capture effect, but without a timing condition. So far, it has only been found to occur with more advanced radios, for example, certain Wi-Fi transceivers, that also perform preamble search after the reception of a valid preamble [91]. Such radios can "abort" an ongoing reception and "switch" to another signal that exceeds the capture threshold, possibly also multiple times. Eventually the strongest of all overlapping signals is received with high probability. For the same physical layer and radio technology, the probability of correctly receiving a message despite a collision is higher than with radios that only feature the capture effect, because it does not matter when the strongest signal arrives. Although a low-power, low-complexity receiver that features the message-in-message effect is yet to be built, we ask:

> *To what extent can the message-in-message effect in combination with a suitable protocol design improve reliability and efficiency compared with protocols relying only on capture?*

Packet-level reliability. To achieve a desired reliability, many synchronous transmissions schemes opportunistically retransmit the same packet [27, 41] or use end-to-end retransmissions [40, 152]. Recent proposals exploit different forms of network coding for higher reliability and efficiency [29, 47, 94]. These techniques operate above the packet level, considering a packet an indivisible unit. Instead, mechanisms such as forward error correction (FEC), interleaving, and exploiting partially correct packets operate at a lower level and are largely unexplored in synchronous transmissions. A standing issue is therefore:

> *What packet-level reliability techniques may complement existing solutions above the packet level when using synchronous transmissions?*

The concrete application of these mechanisms should be informed by physical-layer observations, such as the distribution and pattern of symbol errors in received packets when using synchronous transmissions [6]. Even though further research is necessary in this area too, we conjecture two

possible trends. First, carrier frequency offsets among synchronous transmitters cause a change in the envelope of the received signal, because the receiver cannot simultaneously compensate against all transmitters. Depending on the magnitude of carrier frequency offsets, these envelope changes may translate into characteristic patterns in the sequence of correct and incorrect symbols. Second, one might probably observe a sloping pattern, where symbols toward the end of a packet are more likely to be corrupted than symbols at the beginning of the packet.

If the former trend is confirmed, a protocol could combine packets based on the received signal strength (RSSI): Portions of packets received with high RSSI are likely correct and could be combined to reconstruct the complete, correct packet. This approach exploits the fact that although the error patterns in two corrupt packets may be similar, the patterns are likely shifted so symbols that are corrupt in one packet are likely correct in the other packet, and vice versa.

Further, if the second hypothesis holds, a protocol could let a transmitter alternate between sending the payload bytes in forward and reverse order, so receivers can combine the (likely correct) initial portion of the received packets to reconstruct the complete, correct packet. Partial packet recovery [58] and packet combining [30] are studied in wireless networks, yet they are unexplored in synchronous transmissions, where nodes typically have multiple opportunities to receive the same packet. When using FEC, a protocol could statically or dynamically adjust the size of the code blocks and the amount of added redundancy based on the observed distribution and pattern of symbol errors. FEC would also represent a particularly good fit for synchronous transmissions schemes, where a sender often has no feedback on the successful reception of a packet by a particular node. State-of-the-art 32-bit platforms offer ample compute power to explore the integration of block or convolutional coding into the time-sensitive operation of protocols using synchronous transmissions [100].

Interpreting interfering signals. Protocols and services in this article are specifically designed for physical layers that map the interfering signals at a receiver to *at most* one packet. However, other interpretations of interfering signals are found in the literature, and could be exploited to enhance the capabilities and improve the performance of protocols and services using synchronous transmissions. For example, by interpreting the length and magnitude of the received signal strength, synchronous transmissions have been used for single-hop agreement [13] and counting [148]. Others have looked at disentangling the interfering signals in order to decode *multiple* packets, based on IEEE 802.15.4 [51, 68], LoRa [35], and IEEE 802.11 [44]. While received signal strength interpretations work on commodity low-power wireless platforms, existing multi-packet decoding schemes require special hardware and additional computational power, for example, a software-defined radio or a wall-powered gateway. An open research question is thus:

> *How to best leverage and integrate alternative interpretations of interfering signals into a communication protocol, and what are the end-to-end improvements at scale, for example, in terms of throughput and overall energy costs?*

Recent work on low-power software-defined radios [48] could support this kind of efforts.

5.4 Integration and Standardization

Low-power wireless networks must often integrate with other systems, such as the cloud, through the Internet. One way to fulfill this requirement this is to push characteristic features of the Internet protocol suite, like asynchronous point-to-point interactions, down into low-power wireless networks by means of dedicated designs [67]. Synchronous transmissions prompt a different approach: message exchanges are synchronous and point-to-point interactions are inherently discouraged, as communication tends to be broadcast.

Besides a few works focusing on end-to-end real-time guarantees in CPS applications [55, 57], the interfaces required for a seamless integration are largely unexplored. Therefore, we ask:

> *How are network functionality of a low-power wireless network using synchronous transmissions exposed to the outside world and accessed at runtime?*

and also:

> *How can a globally distributed application specify end-to-end requirements, for example, in reliability and latency, between nodes residing in different low-power wireless networks using synchronous transmissions?*

A stepping stone to solve this issue may be to create a form of decoupling between the required functionality. Existing literature includes a few attempts in this direction; for example, Bolt is dual-processor platform that allows one to isolate the time-sensitive operation of synchronous transmissions from the (a)synchronous operation of applications or other networks and systems [127]. One of the processors runs the network stack using synchronous transmissions, while the other processor implements the application logic.

The question, however, is foremost relevant at the higher levels of the stack, where protocols such as MQTT or CoAP operate. Key assumptions in their designs is that multicast or broadcast communication is costly and that nodes operate asynchronously. A variant of MQTT, called MQTT-SN, accounts for the limited resources and duty-cycled operation of battery-powered devices [118]. On the contrary, synchronous transmissions enable efficient reliable broadcast communication, and requires no knowledge of per-node sleep schedules at an MQTT-SN gateway [118]. Synchronous transmissions may rather be used as building blocks for network stacks that are conceptually closer to field busses, such as CAN or FlexRay. A standing integration issue is therefore:

> *What interfaces and network functionality are most appropriate or required to integrate with real-time networking systems, for example, with Time-Triggered Ethernet or field busses such as CAN or FlexRay?*

We believe that work on integration is essential for widespread adoption and standardization. Close collaboration with industry will be key to identify the relevant platforms, benchmarking scenarios, and standardization practices. Ideally, the identified scenarios would also serve as demonstrators, showing either functionality or performance previously unattained with existing link-based transmissions. We argue that industrial control and autonomous robotics are particularly suitable areas for initial standardization efforts, which would greatly benefit from the end-to-end guarantees and efficient performance in mobile settings enabled by synchronous transmissions [40, 89, 154].

6 CONCLUDING REMARKS

This article surveys communication protocols and network services for low-power wireless systems that exploit the concept of synchronous transmissions. As synchronous transmissions on commodity off-the-shelf devices and existing low-power wireless physical layers become popular just over the past eight years, the field is relatively young, and yet it already witnessed large amounts of work. This article compares the diverse communication protocol designs in a structured way and summarizes the investigated types of network services. Together with our complementary article [6], which provides a tutorial-style introduction to the principles of synchronous transmissions and surveys existing research that aims to better understand or to improve the reliability of the basic technique, we allow newcomers to the field of synchronous transmissions to quickly obtained an overview of the foundations and of the current research art. Although the current state of the art successfully addresses several key requirements of CPS and IoT applications, including dependability, adaptability, and efficiency, further efforts are also required in several directions,

ranging from understanding the implications of physical-layer features on protocol and service designs, to addressing integration and standardization issues.

ACKNOWLEDGMENTS

We thank Carlo Alberto Boano, Carsten Herrmann, Fabian Mager, Johannes Richter, and Romain Jacob for their constructive feedback on earlier drafts of this article. This work was supported in part by the German Research Foundation (DFG) within the Emmy Noether project "NextIoT" (grant ZI 1635/2-1) and by the Swedish Foundation for Strategic Research (SSF) within the project "Future Factories in the Cloud (FiC)" (grant GMT14-0032).

REFERENCES

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. 2000. Network Information Flow. *IEEE Trans. Inf. Theory* 46, 4 (2000).

[2] Jamal N. Al-Karaki and Ahmed E. Kamal. 2004. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Commun.* 11, 6 (2004).

[3] Beshr Al Nahas, Simon Duquennoy, and Olaf Landsiedel. 2017. Network-wide Consensus Utilizing the Capture Effect in Low-power Wireless Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys).*

[4] Beshr Al Nahas, Simon Duquennoy, and Olaf Landsiedel. 2019. Concurrent Transmissions for Multi-Hop Bluetooth 5. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN).*

[5] Noga Alon, Mohsen Ghaffari, Bernhard Haeupler, and Majid Khabbazian. 2014. Broadcast Throughput in Radio Networks: Routing vs. Network Coding. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA).*

[6] Authors. 2020. Title.

[7] Michael Baddeley, Usman Raza, Aleksandar Stanoev, George Oikonomou, Reza Nejabati, Mahesh Sooriyabandara, and Dimitra Simeonidou. 2019. Atomic-SDN: Is Synchronous Flooding the Solution to Software-Defined Networking in IoT? *IEEE Access* 7 (2019).

[8] Dominik Baumann, Fabian Mager, Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. 2019. Fast Feedback Control over Multi-hop Wireless Networks with Mode Changes and Stability Guarantees. *ACM Transactions on Cyber-Physical Systems* 4, 2 (2019).

[9] Albert Benveniste, Paul Caspi, Stephen A. Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone. 2003. The Synchronous Languages 12 Years Later. *Proc. IEEE* 91, 1 (2003).

[10] Naveed Anwar Bhatti, Muhammad Hamad Alizai, Affan A. Syed, and Luca Mottola. 2016. Energy Harvesting and Wireless Transfer in Sensor Network Applications: Concepts and Experiences. *ACM Transactions on Sensor Networks* 12, 3 (2016).

[11] Kenneth P. Birman and Thomas A. Joseph. 1987. Exploiting Virtual Synchrony in Distributed Systems. In *Proc. of the ACM Symposium on Operating Systems Principles (SOSP).*

[12] Carlo Alberto Boano, Thiemo Voigt, Claro Noda, Kay Römer, and Marco Zúñiga. 2011. JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN).*

[13] Carlo Alberto Boano, Marco Antonio Zúñiga, Kay Römer, and Thiemo Voigt. 2012. JAG: Reliable and Predictable Wireless Agreement under External Radio Interference. In *Proc. of the IEEE Real-Time Systems Symposium (RTSS).*

[14] Martin Bor, Utz Roedig, Thiemo Voigt, and Juan M. Alonso. 2016. Do LoRa Low-Power Wide-Area Networks Scale?. In *Proc. of the ACM Int. Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM).*

[15] Martina Brachmann, Olaf Landsiedel, Diana Göhringer, and Silvia Santini. 2019. Whisper: Fast Flooding for Low-Power Wireless Networks. *ACM Trans. Sen. Netw.* 15, 4, Article 47 (Oct. 2019), 26 pages.

[16] Martina Brachmann, Olaf Landsiedel, and Silvia Santini. 2016. Concurrent Transmissions for Communication Protocols in the Internet of Things. In *Proc. of the IEEE Conference on Local Computer Networks (LCN).*

[17] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. 2006. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proc. of the Int. Conference on Embedded Networked Sensor Systems (SENSYS).*

[18] Doug Carlson, Marcus Chang, Andreas Terzis, Yin Chen, and Omprakash Gnawali. 2013. Forwarder Selection in Multi-Transmitter Networks. In *Proc. of the IEEE Int. Conference on Distributed Computing in Sensor Systems (DCOSS).*

[19] Nessrine Chakchouk. 2015. A Survey on Opportunistic Routing in Wireless Communication Networks. *IEEE Communications Surveys Tutorials* 17, 4 (2015).

[20] Tengfei Chang, Thomas Watteyne, Xavier Vilajosana, and Pedro Henrique Gomes. 2019. Constructive Interference in 802.15.4: A Tutorial. *IEEE Commun. Surveys Tuts.* 21, 1 (2019).

[21] Pablo Corbalán and Gian Pietro Picco. 2018. Concurrent Ranging in Ultra-wideband Radios: Experimental Evidence, Challenges, and Opportunities.. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[22] Pablo Corbalán, Gian Pietro Picco, and Sameera Palipana. 2019. Chorus: UWB Concurrent Transmissions for GPS-like Passive Localization of Countless Targets. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[23] Carlos de Morais Cordeiro, Dharma P. Agrawal, and Djamel Hadj Sadok. 2003. Interference Modeling and Performance of Bluetooth MAC Protocol. *IEEE Trans. Wireless Commun.* 2, 6 (2003).
[24] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. 2011. *Distributed Systems: Concepts and Design* (5th ed.). Addison-Wesley Publishing Company, USA.
[25] Ilker Demirkol, Cem Ersoy, and Fatih Alagöz. 2006. MAC Protocols for Wireless Sensor Networks: A Survey. *IEEE Commun. Mag.* 44, 4 (2006).
[26] Manjunath Doddavenkatappa and Mun Choon Chan. 2014. P^3: A Practical Packet Pipeline Using Synchronous Transmissions for Wireless Sensor Networks. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[27] Manjunath Doddavenkatappa, Mun Choon Chan, and Ben Leong. 2013. Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.
[28] Wan Du, Jansen Christian Liando, Huanle Zhang, and Mo Li. 2015. When Pipelines Meet Fountain: Fast Data Dissemination in Wireless Sensor Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[29] Wan Du, Jansen Christian Liando, Huanle Zhang, and Mo Li. 2017. Pando: Fountain-Enabled Fast Data Dissemination With Constructive Interference. *IEEE/ACM Trans. Netw.* 25, 2 (2017).
[30] Henri Dubois-Ferrière, Deborah Estrin, and Martin Vetterli. 2005. Packet Combining in Sensor Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[31] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. 2010. Design and Evaluation of a Versatile and Efficient Receiver-initiated Link Layer for Low-power Wireless. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[32] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. 2012. A-MAC: A Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. *ACM Transactions on Sensor Networks* 8, 4 (2012).
[33] Prabal Dutta, Razvan Musaloiu-E., Ion Stoica, and Andreas Terzis. 2008. Wireless ACK Collisions Not Considered Harmful. In *Proc. of the ACM Workshop on Hot Topics in Networks (HotNets)*.
[34] Prabal K. Dutta and David E. Culler. 2005. System Software Techniques for Low-Power Operation in Wireless Sensor Networks. In *Proc. of the IEEE/ACM Int. Conference on Computer-aided Design (ICCAD)*.
[35] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yağan. 2017. Empowering Low-Power Wide Area Networks in Urban Settings. In *Proc. of ACM SIGCOMM*.
[36] Jeremy Elson and Kay Römer. 2003. Wireless Sensor Networks: A New Regime for Time Synchronization. *ACM SIGCOMM Computer Communication Review* 33, 1 (2003).
[37] Antonio Escobar, Cristian Gonzalez, Francisco J. Cruz, Javier Garcia-Jimenez, Jirka Klaue, and Angel Corona. 2016. RedFixHop: Efficient Ultra-Low-Latency Network Flooding. In *Proc. of the IEEE Int. Conference on Sensing, Communication, and Networking (SECON)*.
[38] Antonio Escobar-Molero. 2019. Improving Reliability and Latency of Wireless Sensor Networks using Concurrent Transmissions. *Automatisierungstechnik* 67, 1 (2019).
[39] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. 2012. Low-Power Wireless Bus. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[40] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. 2013. Virtual Synchrony Guarantees for Cyber-Physical Systems. In *Proc. of the IEEE Int. Symposium on Reliable Distributed Systems (SRDS)*.
[41] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. 2011. Efficient Network Flooding and Time Synchronization with Glossy. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[42] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer. 2006. Network Coding: An Instant Primer. *ACM SIGCOMM Computer Communication Review* 36, 1 (2006).
[43] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. 2009. Collection Tree Protocol. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[44] Shyamnath Gollakota and Dina Katabi. 2008. ZigZag Decoding: Combating Hidden Terminals in Wireless Networks. In *Proc. of ACM SIGCOMM*.

[45] Bernhard Großwindhager, Michael Stocker, Michael Rath, Carlo Alberto Boano, and Kay Römer. 2019. SnapLoc: An Ultra-Fast UWB-Based Indoor Localization System for an Unlimited Number of Tags. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[46] Bernhard Haeupler. 2016. Analyzing Network Coding (Gossip) Made Easy. *J. ACM* 63, 3 (2016).
[47] Carsten Herrmann, Fabian Mager, and Marco Zimmerling. 2018. Mixer: Efficient Many-to-All Broadcast in Dynamic Wireless Mesh Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[48] Mehrdard Hessar, Ali Najafi, Vikram Iyer, and Shyamnath Gollakota. 2020. TinySDR: Low-Power SDR Platform for Over-the-Air Programmable IoT Testbeds. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.
[49] Tracey Ho, Muriel Médard, Ralf Koetter, David R. Karger, Michelle Effros, Jun Shi, and Ben Leong. 2006. A Random Linear Network Coding Approach to Multicast. *IEEE Trans. Inf. Theory* 52, 10 (2006).
[50] Jonathan W. Hui and David E. Culler. 2008. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[51] Dali Ismail, Mahbubur Rahman, Abusayeed Saifullah, and Sanjay Madria. 2017. RnR: Reverse & Replace Decoding for Collision Recovery in Wireless Sensor Networks. In *Proc. of the IEEE Int. Conference on Sensing, Communication, and Networking (SECON)*.
[52] Timofei Istomin, Amy L. Murphy, Gian Pietro Picco, and Usman Raza. 2016. Data Prediction + Synchronous Transmissions = Ultra-low Power Wireless Sensor Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[53] Timofei Istomin, Matteo Trobinger, Amy L. Murphy, and Gian Pietro Picco. 2018. Interference-Resilient Ultra-Low Power Aperiodic Data Collection. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[54] Romain Jacob, Jonas Baechli, Reto Da Forno, and Lothar Thiele. 2019. Synchronous Transmissions Made Easy: Design Your Network Stack with Baloo. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[55] Romain Jacob, Licong Zhang, Marco Zimmerling, Jan Beutel, Samarjit Chakraborty, and Lothar Thiele. 2018. TTW: A Time-Triggered Wireless Design for CPS. In *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*.
[56] Romain Jacob, Licong Zhang, Marco Zimmerling, Samarjit Chakraborty, and Lothar Thiele. 2020. The Time-Triggered Wireless Architecture. In *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*.
[57] Romain Jacob, Marco Zimmerling, Pengcheng Huang, Jan Beutel, and Lothar Thiele. 2016. End-to-end Real-time Guarantees in Wireless Cyber-physical Systems. In *Proc. of the IEEE Real-Time Systems Symposium (RTSS)*.
[58] Kyle Jamieson and Hari Balakrishnan. 2007. PPR: Partial Packet Recovery for Wireless Networks. In *Proc. of ACM SIGCOMM*.
[59] Kyle Jamieson, Hari Balakrishnan, and Y. C. Tay. 2006. Sift: A MAC Protocol for Event-Driven Wireless Sensor Networks. In *Proc. of the European Workshop on Wireless Sensor Networks (EWSN)*.
[60] Farhana Javed, Muhamamd Khalil Afzal, Muhammad Sharif, and Byung-Seo Kim. 2018. Internet of Things (IoT) Operating Systems Support, Networking Technologies, Applications, and Challenges: A Comparative Review. *IEEE Commun. Surveys Tuts.* 20, 3 (2018).
[61] Jongsoo Jeong, Jongjun Park, Hoon Jeong, JongArm Jun, Mike Chieh-Jan Liang, and JeongGil Ko. 2014. Low-Power and Topology-Free Data Transfer Protocol with Synchronous Packet Transmissions. In *Proc. of the IEEE Int. Conference on Sensing, Communication, and Networking (SECON)*.
[62] Xiaoyu Ji, Yuan He, Jiliang Wang, Wei Dong, Xiaopei Wu, and Yunhao Liu. 2014. Walking down the STAIRS: Efficient Collision Resolution for Wireless Sensor Networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*.
[63] Jens Karschau, Marco Zimmerling, and Benjamin M. Friedrich. 2018. Renormalization Group Theory for Percolation in Time-varying Networks. *Scientific Reports* 8 (2018).
[64] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. 2006. XORs in The Air: Practical Wireless Network Coding. In *Proc. of ACM SIGCOMM*.
[65] Benjamin Kempke, Pat Pannuto, Bradford Campbell, and Prabal Dutta. 2016. SurePoint: Exploiting Ultra Wideband Flooding and Diversity to Provide Robust, Scalable, High-Fidelity Indoor Localization. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[66] Johannes Klinglmayr, Christian Bettstetter, Marc Timme, and Christoph Kirst. 2017. Convergence of Self-Organizing Pulse-Coupled Oscillator Synchronization in Dynamic Networks. *IEEE Trans. Autom. Control* 62, 4 (2017).
[67] JeongGil Ko, Andreas Terzis, Stephen Dawson-Haggerty, David E. Culler, Jonathan W. Hui, and Philip Levis. 2011. Connecting Low-Power and Lossy Networks to the Internet. *IEEE Commun. Mag.* 49, 4 (2011).
[68] Linghe Kong and Xue Liu. 2015. mZig: Enabling Multi-Packet Reception in ZigBee. In *Proc. of the ACM Int. Conference on Mobile Computing and Networking (MobiCom)*.

[69] Michael König and Roger Wattenhofer. 2016. Maintaining Constructive Interference Using Well-Synchronized Sensor Nodes. In *Proc. of the IEEE Int. Conference on Distributed Computing in Sensor Systems (DCOSS)*.
[70] Nupur Kothari, Todd Millstein, and Ramesh Govindan. 2008. Deriving State Machines from TinyOS Programs Using Symbolic Execution. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[71] Olaf Landsiedel, Federico Ferrari, and Marco Zimmerling. 2013. Chaos: Versatile and Efficient All-to-All Data Sharing and In-Network Processing at Scale. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[72] J. Nicholas Laneman, David N. C. Tse, and Gregory W. Wornell. 2004. Cooperative Diversity in Wireless Networks: Efficient Protocols and Outage Behavior. *IEEE Trans. Inf. Theory* 50, 12 (2004).
[73] Edward A. Lee. 2008. Cyber Physical Systems: Design Challenges. In *Proc. of the IEEE Int. Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*.
[74] Jeongkeun Lee, Wonho Kim, Sung-Ju Lee, Daehyung Jo, Jiho Ryu, Taekyoung Kwon, and Yanghee Choi. 2007. An Experimental Study on the Capture Effect in 802.11a Networks. In *Proc. of the ACM Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*.
[75] Krijn Leentvaar and Jan H. Flint. 1976. The Capture Effect in FM Receivers. *IEEE Trans. Commun.* (1976).
[76] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. 2009. Optimal Clock Synchronization in Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[77] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. 2015. PulseSync: An Efficient and Scalable Clock Synchronization Protocol. *IEEE/ACM Trans. Netw.* 23, 3 (2015).
[78] Philip Levis, Sam Madden, David Gay, Joseph Polastre, Robert Szewczyk, Alec Woo, Eric Brewer, Eric Brewer, Eric Brewer, and David Culler. 2004. The Emergence of Networking Abstractions and Techniques in TinyOS. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.
[79] C. Liao, G. Zhu, D. Kuwabara, M. Suzuki, and H. Morikawa. 2017. Multi-Hop LoRa Networks Enabled by Concurrent Transmission. *IEEE Access* 5 (2017), 21430–21446.
[80] Chun-Hao Liao, Yuki Katsumata, Makoto Suzuki, and Hiroyuki Morikawa. 2016. Revisiting the So-called Constructive Interference in Concurrent Transmission. In *Proc. of the IEEE Conference on Local Computer Networks (LCN)*.
[81] Roman Lim, Reto Da Forno, Felix Sutton, and Lothar Thiele. 2017. Competition: Robust Flooding Using Back-to-Back Synchronous Transmissions with Channel-Hopping. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[82] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. 2013. FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[83] Roman Lim, Balz Maag, and Lothar Thiele. 2016. Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[84] Daibo Liu, Zhichao Cao, and Mengshu Hou. 2019. ALIGNER: Make the Utmost of Transmission Concurrency for Low Power Wireless Networks. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[85] Ye Liu, Qi Chen, Hao Liu, Chen Hu, and Qing Yang. 2016. A Non Destructive Interference Based Receiver-Initiated MAC Protocol for Wireless Sensor Networks. In *Proc. of the IEEE Annual Consumer Communications & Networking Conference (CCNC)*.
[86] Diego Lobba, Matteo Trobinger, Davide Vecchia, Timofei Istomin, and Gian Pietro Picco. 2012. Concurrent Transmissions for Multi-hop Communication on Ultra-wideband Radios. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[87] Xiaoyuan Ma, Peiling Zhang, Carlo Alberto Boano, Ye Liu, Jun Huang, and Jianming Wei. 2020. Harmony: Saving Concurrent Transmissions from Harsh RF Interference. In *Proc. of the IEEE Int. Conference on Computer Communications (INFOCOM)*.
[88] Xiaoyuan Ma, Peilin Zhang, Oliver Theel, and Jianming Wei. 2020. Gathering data with packet-in-packet in wireless sensor networks. *Computer Networks* 170 (2020).
[89] Fabian Mager, Dominik Baumann, Romain Jacob, Lothar Thiele, Sebastian Trimpe, and Marco Zimmerling. 2019. Feedback Control Goes Wireless: Guaranteed Stability over Low-power Multi-hop Networks. In *Proc. of the ACM/IEEE Int. Conference on Cyber-Physical Systems (ICCPS)*.
[90] Fabian Mager, Carsten Herrmann, and Marco Zimmerling. 2017. One for All, All for One: Toward Efficient Many-to-Many Broadcast in Dynamic Wireless Networks. In *Proc. of the ACM Workshop on Hot Topics in Wireless (HotWireless)*.
[91] Justin Manweiler, Naveen Santhapuri, Souvik Sen, Romit Roy Choudhury, Srihari Nelakuditi, and Kamesh Munagala. 2009. Order Matters: Transmission Reordering in Wireless Networks. In *Proc. of the ACM Int. Conference on Mobile Computing and Networking (MobiCom)*.
[92] Guoqiang Mao, Barış Fidan, and Brian D.O. Anderson. 2007. Wireless sensor network localization techniques. *Computer Networks* 51, 10 (2007).

[93] Ivan Minakov, Roberto Passerone, Alessandra Rizzardi, and Sabrina Sicari. 2016. A Comparative Study of Recent Wireless Sensor Network Simulators. *ACM Transactions on Sensor Networks* 12, 3 (2016).
[94] Mobashir Mohammad and Mun Choon Chan. 2018. Codecast: Supporting Data Driven In-Network Processing for Low-Power Wireless Sensor Networks. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[95] Luca Mottola and Gian Pietro Picco. 2011. Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art. *Comput. Surveys* 43, 3 (2011).
[96] Luca Mottola and Gian Pietro Picco. 2012. Middleware for wireless sensor networks: an outlook. *Journal of Internet Services and Applications* 3, 1 (2012).
[97] Luca Mottola, Gian Pietro Picco, Felix Jonathan Opperman, Joakim Eriksson, Niclas Finne, Harald Fuchs, Andrea Gaglione, Stamatis Karnouskos, Patricio Montero, Nina Oertel, Kay Römer, Patrik Spieß, Stefano Tranquillini, and Thiemo Voigt. 2017. makeSense: Simplifying the Integration of Wireless Sensor Networks into Business Processes. *IEEE Trans. Softw. Eng.* 45, 6 (2017).
[98] Razvan Musaloiu-E, Chieh-Jan Mike Liang, and Andreas Terzis. 2008. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proc. of the Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[99] Fredrik Österlind, Luca Mottola, Thiemo Voigt, Nicolas Tsiftes, and Adam Dunkels. 2012. Strawman: Resolving Collisions in Bursty Low-Power Wireless Networks. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.
[100] Jongjun Park, Jongsoo Jeong, Hoon Jeong, Chieh-Jan Mike Liang, and JeongGil Ko. 2014. Improving the Packet Delivery Performance for Concurrent Packet Transmissions in WSNs. *IEEE Commun. Lett.* 18, 1 (2014).
[101] Valentin Poirot, Beshr Al Nahas, and Olaf Landsiedel. 2019. Paxos Made Wireless: Consensus in the Air. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[102] Alexandros Pollakis, Lucas Wetzel, David J. Jörg, Wolfgang Rave, Gerhard Fettweis, and Frank Jülicher. 2014. Synchronization in networks of mutually delay-coupled phase-locked loops. *New Journal of Physics* 16, 11 (2014).
[103] James A. Preiss, Wolfgang Hönig, Gaurav S. Sukhatme, and Nora Ayanian. 2017. Crazyswarm: A Large Nano-Quadcopter Swarm. In *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*.
[104] Hariharan Rahul, Haitham Hassanieh, and Dina Katabi. 2010. SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity. In *Proc. of ACM SIGCOMM*.
[105] Bhaskaran Raman, Kameswari Chebrolu, Sagar Bijwe, and Vijay Gabale. 2010. PIP: A Connection-Oriented, Multi-Hop, Multi-Channel TDMA-based MAC for High Throughput Bulk Transfer. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[106] Usman Raza, Alessandro Camerra, Amy L. Murphy, Themis Palpanas, and Gian Pietro Picco. 2015. Practical Data Prediction for Real-World Wireless Sensor Networks. *IEEE Trans. Knowl. Data Eng.* 27, 8 (2015).
[107] Coen Roest. 2015. *Enabling the Chaos Networking Primitive on Bluetooth LE.* Master's thesis. Delft University of Technology.
[108] Sudipta Saha and Mun Choon Chan. 2017. Design and Application of a Many-to-One Communication Protocol. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*.
[109] Yuya Saito, Anass Benjebbour, Yoshihisa Kishiyama, and Takehiro Nakamura. 2013. System-Level Performance Evaluation of Downlink Non-orthogonal Multiple Access (NOMA). In *Proc. of the IEEE Int. Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*.
[110] Silvia Santini and Kay Römer. 2006. An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks. In *Proceedings of the Int. Conference on Networked Sensing Systems (INSS)*.
[111] C. Sarkar, R. V. Prasad, and K. Langendoen. 2019. FLEET: When Time-Bounded Communication Meets High Energy-Efficiency. *IEEE Access* 7 (2019).
[112] Chayan Sarkar, R. Venkatesha Prasad, Raj Thilak Rajan, and Koen Langendoen. 2016. Sleeping Beauty: Efficient Communication for Node Scheduling. In *Proc. of the IEEE Int. Conference on Mobile Ad Hoc and Sensor Systems (MASS)*.
[113] Fred B. Schneider. 1990. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *Comput. Surveys* 22, 4 (1990).
[114] Markus Schuß, Carlo Alberto Boano, Manuel Weber, and Kay Römer. 2017. A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN)*.
[115] Andrew Sendonaris, Elza Erkip, and Behnaam Aazhang. 2003. User Cooperation Diversity—Part I: System Description. *IEEE Trans. Commun.* 51, 11 (2003).
[116] Andrew Sendonaris, Elza Erkip, and Behnaam Aazhang. 2003. User Cooperation Diversity—Part II: Implementation Aspects and Performance Analysis. *IEEE Trans. Commun.* 51, 11 (2003).
[117] Alberto Spina, Michael Breza, Naranker Dulay, and Julie McCann. 2020. XPC: Fast and Reliable Synchronous Transmission Protocols for 2-Phase Commit and 3-Phase Commit. In *Proc. of the Int. Conference on Embedded Wireless*

Systems and Networks (EWSN).

[118] Andy Stanford-Clark and Hong Linh Truong. 2013. MQTT For Sensor Networks (MQTT-SN) Protocol Specification. **http://www.mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf**

[119] John A. Stankovic. 2014. Research Directions for the Internet of Things. *IEEE Internet Things J.* 1, 1 (2014).

[120] John A. Stankovic, Tarek F. Abdelzaher, Chenyang Lu, and Jennifer C. Sha, Lui Hou. 2003. Real-Time Communication and Coordination in Embedded Sensor Networks. *Proc. IEEE* 91, 7 (2003).

[121] John A. Stankovic, Insup Lee, Aloysius Mok, and Raj Rajkumar. 2005. Opportunities and Obligations for Physical Computing Systems. *Computer* 38, 11 (2005).

[122] Yanjun Sun, Omer Gurewitz, and David B. Johnson. 2008. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys).*

[123] Felix Sutton, Bernhard Buchli, Jan Beutel, and Lothar Thiele. 2015. Zippy: On-Demand Network Flooding. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys).*

[124] Felix Sutton, Reto Da Forno, Jan Beutel, and Lothar Thiele. 2017. BLITZ: A Network Architecture for Low Latency and Energy-efficient Event-triggered Wireless Communication. In *Proc. of the ACM Workshop on Hot Topics in Wireless (HotWireless).*

[125] Felix Sutton, Reto Da Forno, Jan Beutel, and Lothar Thiele. 2019. BLITZ: Low Latency and Energy-Efficient Communication for Event-Triggered Wireless Sensing Systems. *ACM Transactions on Sensor Networks* 15, 2 (2019).

[126] Felix Sutton, Reto Da Forno, David Gschwend, Tonio Gsell, Roman Lim, Jan Beutel, and Lothar Thiele. 2017. The Design of a Responsive and Energy-efficient Event-triggered Wireless Sensing System. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN).*

[127] Felix Sutton, Marco Zimmerling, Reto Da Forno, Roman Lim, Tonio Gsell, Georgia Giannopoulou, Federico Ferrari, Jan Beutel, and Lothar Thiele. 2015. Bolt: A Stateful Processor Interconnect. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15).*

[128] Makoto Suzuki, Chun-Hao Liao, Sotaro Ohara, Kyoichi Jinno, and Hiroyuki Morikawa. 2017. Wireless-Transparent Sensing. In *Proc. of the Int. Conference on Embedded Wireless Systems and Networks (EWSN).*

[129] Makoto Suzuki, Yasutaka Yamashita, and Hiroyuki Morikawa. 2013. Low-Power, End-to-End Reliable Collection using Glossy for Wireless Sensor Networks. In *Proc. of the IEEE Vehicular Technology Conference (VTC Spring).*

[130] Federico Terraneo, Alberto Leva, Silvano Seva, Martina Maggio, and Alessandro Vittorio Papadopoulos. 2015. Reverse Flooding: exploiting radio interference for efficient propagation delay compensation in WSN clock synchronization. In *Proc. of the IEEE Real-Time Systems Symposium (RTSS).*

[131] Christian Tschudin. 2019. A Broadcast-Only Communication Model Based on Replicated Append-Only Logs. *ACM SIGCOMM Computer Communication Review* 49, 2 (2019).

[132] David Tse and Pramod Viswanath. 2005. *Fundamentals of Wireless Communication.* Cambridge University Press, New York, NY, USA.

[133] Davide Vecchia, Pablo Corbalán, Timofei Istomin, and Gian Pietro Picco. 2019. Playing with Fire: Exploring Concurrent Transmissions in Ultra-wideband Radios. In *Proc. of the IEEE Int. Conference on Sensing, Communication and Networking (SECON).*

[134] Sergio Verdu. 1998. *Multiuser Detection.* Cambridge University Press, New York, NY, USA.

[135] Yin Wang, Yuan He, Dapeng Cheng, Yunhao Liu, and Xiang-yang Li. 2013. TriggerCast: Enabling Wireless Constructive Collisions. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM).*

[136] Yin Wang, Yuan He, Xufei Mao, Yunhao Liu, Zhiyu Huang, and Xiangyang Li. 2012. Exploiting Constructive Interference for Scalable Flooding in Wireless Networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM).*

[137] Yin Wang, Yuan He, Xufei Mao, Yunhao Liu, and Xiang-yang Li. 2013. Exploiting Constructive Interference for Scalable Flooding in Wireless Networks. *IEEE/ACM Trans. Netw.* 21, 6 (2013).

[138] Yin Wang, Yunhao Liu, Yuan He, Xiang-Yang Li, and Dapeng Cheng. 2015. Disco: Improving Packet Delivery via Deliberate Synchronized Constructive Interference. *IEEE Trans. Parallel Distrib. Syst.* 26, 3 (2015).

[139] Yin Wang, Gaofeng Pan, and Zhiyu Huang. 2012. Poster Abstract: Direct Multi-hop Time Synchronization with Constructive Interference. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN).*

[140] Kamin Whitehouse, Alec Woo, Fred Jiang, Joseph Polastre, and David Culler. 2005. Exploiting The Capture Effect For Collision Detection And Recovery. In *Proc. of the IEEE Workshop on Embedded Networked Sensors (EmNetS).*

[141] Matthias Wilhelm, Vincent Lenders, and Jens B. Schmitt. 2014. On the Reception of Concurrent Transmissions in Wireless Sensor Networks. *IEEE Trans. Wireless Commun.* 13, 12 (2014).

[142] Ebram Kamal William and Mun Choon Chan. 2019. InDP: In-Network Data Processing for Wireless Sensor Networks. In *Proc. of the IEEE Int. Conference on Sensing, Communication, and Networking (SECON).*

[143] Dingming Wu, Chao Dong, Shaojie Tang, Haipeng Dai, and Guihai Chen. 2014. Fast and Fine-grained Counting and Identification via Constructive Interference in WSNs. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.

[144] Shuying Yu, Xiaobing Wu, Pan Wu, Dingming Wu, Haipeng Dai, and Guihai Chen. 2014. CIRF: Constructive Interference-based Reliable Flooding in Asynchronous Duty-Cycle Wireless Sensor Networks. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*.

[145] Dingwen Yuan and Matthias Hollick. 2013. Let's Talk Together: Understanding Concurrent Transmission in Wireless Sensor Networks. In *Proc. of the IEEE Conference on Local Computer Networks (LCN)*.

[146] Dingwen Yuan and Matthias Hollick. 2015. Ripple: High-throughput, Reliable and Energy-efficient Network Flooding in Wireless Sensor Networks. In *Proc. of the IEEE Int. Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*.

[147] Dingwen Yuan, Michael Riecker, and Matthias Hollick. 2014. Making 'Glossy' Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-low Latency Communication in Wireless Control Networks. In *Proc. of the European Conference on Wireless Sensor Networks (EWSN)*.

[148] Wenjie Zeng, Anish Arora, and Kannan Srinivasan. 2013. Low Power Counting via Collaborative Wireless Communications. In *Proc. of the ACM/IEEE Int. Conference on Information Processing in Sensor Networks (IPSN)*.

[149] Jin Zhang, Andreas Reinhardt, Wen Hu, and Salil S. Kanhere. 2015. RFT: Identifying Suitable Neighbors for Concurrent Transmissions in Point-to-Point Communications. In *Proc. of the ACM Int. Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*.

[150] Peilin Zhang, Alex Yuan Gao, and Oliver Theel. 2017. Less is More: Learning More with Concurrent Transmissions for Energy-Efficient Flooding. In *Proc. of the 14th EAI Int. Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*.

[151] Peilin Zhang, Xiaoyuan Ma, Oliver Theel, and Jianming Wei. 2018. Concurrent Transmission-based Packet Concatenation in Wireless Sensor Networks. In *Proc. of the IEEE Conference on Local Computer Networks (LCN)*.

[152] Marco Zimmerling, Federico Ferrari, Luca Mottola, and Lothar Thiele. 2013. On Modeling Low-Power Wireless Protocols Based on Synchronous Packet Transmissions. In *Proc. of the IEEE Int. Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*.

[153] Marco Zimmerling, Federico Ferrari, Luca Mottola, Thiemo Voigt, and Lothar Thiele. 2012. pTunes: Runtime parameter adaptation for low-power MAC protocols. In *Proc. of the Int. Conference on Information Processing in Sensor Networks (IPSN)*.

[154] Marco Zimmerling, Luca Mottola, Pratyush Kumar, Federico Ferrari, and Lothar Thiele. 2017. Adaptive Real-Time Communication for Wireless Cyber-Physical Systems. *ACM Transactions on Cyber-Physical Systems* 1, 2 (2017).