Harmony: Saving Concurrent Transmissions from Harsh RF Interference

Xiaoyuan Ma*†, Peilin Zhang‡, Ye Liu§, Carlo Alberto Boano¶, Hyung-Sin Kim‖,
Jianming Wei*, ✉ Jun Huang*
*Shanghai Advanced Research Institute, CAS, China †University of Chinese Academy of Sciences, China
‡Carl von Ossietzky University of Oldenburg, Germany §Nanjing Agricultural University, China
¶Graz University of Technology, Austria ‖Google Inc., USA
{maxy, wjm, huangj}@sari.ac.cn, peilin.zhang@informatik.uni-oldenburg.de,
yeliu@njau.edu.cn, cboano@tugraz.at, hyungkim@google.com

***Abstract*—The increasing congestion of the RF spectrum is a key challenge for low-power wireless networks using concurrent transmissions. The presence of radio interference can indeed undermine their dependability, as they rely on a tight synchronization and incur a significant overhead to overcome packet loss. In this paper, we present *Harmony*, a new data collection protocol that exploits the benefits of concurrent transmissions and embeds techniques to ensure a reliable and timely packet delivery despite highly congested channels. Such techniques include, among others, a data freezing mechanism that allows to successfully deliver data in a partitioned network as well as the use of network coding to shorten the length of packets and increase the robustness to unreliable links. *Harmony* also introduces a distributed interference detection scheme that allows each node to activate various interference mitigation techniques only when strictly necessary, avoiding unnecessary energy expenditures while finding a good balance between reliability and timeliness. An experimental evaluation on real-world testbeds shows that *Harmony* outperforms state-of-the-art protocols in the presence of harsh Wi-Fi interference, with up to 50% higher delivery rates and significantly shorter end-to-end latencies, even when transmitting large packets.**

I. Introduction

Low-power wireless networks are a fundamental building block of the Internet of Things (IoT) and empower valuable applications in various fields [1]–[5]. Many of these applications are mission-critical and pose strict *dependability* requirements on network performance. As a result, low-power wireless networks are often expected to deliver information reliably through multi-hop links within bounded delays, in order to interact effectively with the physical world. These requirements are especially stringent in the context of cyber-physical systems (CPS), where sensor data and actuation commands must be delivered reliably and timely, while the availability of battery-powered devices needs to be maximized.

Such need for dependable communications has challenged the low-power wireless community in the past decade. Whilst conventional protocols are built on the assumption that packet collisions should be avoided, a new wave of solutions based on the concept of *concurrent transmissions* (CT) disputes this view and represents a major technical breakthrough in the development of dependable multi-hop wireless protocols [6].

This work was supported partially by the National Key R&D Program of China (No. 2017YFE0119300), the National Natural Science Foundation of China (No. 61902188), and the German Research Training Group DFG-GRK 1765: System Correctness under Adverse Conditions (SCARE). This work was done when Hyung-Sin Kim was at UC Berkeley.

When using CT, a node can successfully receive a packet purposely transmitted by multiple nodes at the same time on the same carrier frequency, thanks to the capture effect (with timing errors < 160 μs) and constructive interference (with timing errors < 0.5 μs) [7], [8]. Hence, CT allow low-power wireless devices to receive a packet despite a collision, which enables the creation of dependable communication protocols.

Indeed, by having nodes re-broadcast a packet immediately, one can quickly flood information throughout a network without wasting time and energy for routing, scheduling, and other collision avoidance mechanisms. Furthermore, CT allow to use many (if not all) nodes in the network and exploit the spatial diversity to maximize the number of packets successfully delivered despite the presence of unreliable links.

A large number of studies have recognized the potential of CT and proposed low-power wireless protocols exploiting the inherent spatial and temporal redundancy of CT-based flooding to increase the robustness and efficiency of data collection and dissemination [8]–[20]. These works have also shown how to improve the performance of CT-based protocols by using different flooding rounds [8]–[10], retransmission mechanisms [11]–[13], as well as by introducing network coding [14]–[17] and channel hopping techniques [18]–[20].

Challenges. Despite the better performance and versatility compared to traditional schemes, there are major concerns about the practicality of CT-based solutions in real-world settings. Specifically, all the advantages of CT rely on a tight, network-wide time synchronization, i.e., all nodes need to be precisely synchronized to a single *host* (also called initiator). As soon as nodes start losing synchronization to the host, the performance of CT-based networks degrades significantly [20].

Whilst maintaining a tight synchronization for an extended period of time is a challenge in itself [21], it is extremely difficult in the presence of *radio interference*. Indeed, RF interference often leads to a network partition, i.e., to the disconnection of a fraction of nodes from the rest of the network, which makes it hard to keep nodes synchronized.

The presence of radio interference in the surroundings of a low-power wireless network also results in an increased packet loss, end-to-end delay, and energy expenditure [22]. This can be a severe problem in real-world settings, due to the increasing proliferation of wireless devices and consequent congestion of the RF spectrum. For example, the ubiquitousness of Wi-Fi appliances is a major threat to co-located low-

power wireless networks making use of the 2.4 GHz ISM band. Wi-Fi devices, indeed, make use of high data rates (up to 450 Mbps) and large bandwidth channels (20–40 MHz). As a result, the presence of several Wi-Fi networks can lead to a complete congestion of the frequencies used by co-located IEEE 802.15.4 devices [23], which cannot be fully mitigated by CT approaches based only on channel hopping [18]–[20].

To date, the research community did not focus on the design of a CT-based data collection protocol that can achieve a dependable performance in the presence of harsh RF interference, i.e., when all available channels are highly congested.

Closing this gap requires the design of a CT solution that embeds (i) techniques ensuring a reliable and timely packet delivery despite highly congested channels, as well as (ii) the ability to keep time synchronization even when the network is temporarily partitioned. Such a solution should also be able to (iii) sustain a *balanced* performance over time, i.e., enable a reliable and timely data collection without incurring unnecessary energy expenditures. Doing so requires (iv) interference detection schemes allowing each node to quickly quantify the interference in the whole network and decide the corresponding mitigation techniques. This way, nodes can use only the technique(s) necessary to mitigate the interference present at a given point in time, maximizing energy efficiency.

Our approach. In this paper we design *Harmony*, a data collection protocol based on CT that embeds all these features and sustains a dependable performance despite harsh RF interference. *Harmony* divides time into *epochs* and each epoch contains multiple *negotiation-and-action* phases. These are used to differentiate the behaviour of each node in the network over time. During the negotiation, all nodes share their current status (e.g., have data to transmit, data has been received successfully) and determine their next actions (e.g., flood a packet, forward incoming packets, or enter sleep mode) based on the shared information and simple local rules.

To avoid congested frequencies, *Harmony* makes use of channel hopping. In particular, it embeds a *scan-and-lock* mechanism inspired from [20], where each node scans the available channels before each negotiation/action and locks to the least noisy one. To increase the reliability of CT even further, *Harmony* can make use of *network coding* to shorten the length of packets and overcome the packet loss caused by interference. *Harmony* also embeds a novel *data freezing* mechanism enabling a successful data delivery in a partitioned network. To this end, relay nodes can act as source and initiate the flooding of packets received in previous rounds.

The use of these techniques causes a higher energy expenditure in order to deliver packets as reliably and quickly as possible despite interference. Hence, using these techniques all the time (e.g., when interference is absent) would lead to a major energy waste and may even be counter-productive, e.g., cause longer delays. To trigger the usage of one or more features only when strictly necessary, *Harmony* uses a novel interference detection scheme named *SNI*. The latter allows each node to build a bitmap capturing the interference throughout the *whole* network during the negotiation. This bitmap is used to cope with the dynamicity of interference and *balance* reliability, timeliness, and efficiency at runtime.

To maintain time synchronization in the presence of interference, *Harmony* makes use of one or more *synchronization agents*. When a node loses connectivity with the host, i.e., the network is temporarily partitioned, it can locally synchronize with nearby agents and operate normally, rather than impatiently triggering a re-joining procedure [20]. This allows some nodes to deliver packets for an extended time, during which synchronization can be restored once interference has ceased.

We implement *Harmony* on off-the-shelf IEEE 802.15.4 nodes using Contiki [24] and evaluate its performance on two popular testbeds in the presence of artificial RF interference generated using JamLab [25] and JamLab-NG [23]. In particular, we first show that the interference mitigation techniques embedded in *Harmony* are effective and enable a dependable data collection even in the presence of harsh Wi-Fi interference. We then compare the performance of *Harmony* to state-of-the-art CT-based protocols including Mixer [17] and $\mathrm{Crystal}^{\mathrm{CH}}_{\mathrm{ND}}$ [19]. Our results show that *Harmony* outperforms the state-of-the-art in the presence of interference, and that it sustains a high reliability even when transmitting large packets.

Contributions. After describing the key challenges in achieving a dependable data collection under harsh RF interference in Sect. II, this paper makes the following contributions:

- We design and implement *Harmony*, a novel CT-based data collection protocol that sustains a dependable performance even in harsh RF environments (Sect. III).
- We illustrate *Harmony*'s basic components, introducing a novel *negotiation-and-action* phase to schedule each node's behaviour in a distributed manner (Sect. III-A).
- We enrich *Harmony* with two techniques increasing the chances of successful packet delivery in harsh RF environments. These techniques are based on network coding and a novel *data freezing* mechanism ensuring a successful data delivery in a partitioned network (Sect. III-B).
- We present a network-wide interference detection scheme helping *Harmony* to sustain a good balance between reliability, timeliness, and energy efficiency (Sect. III-C).
- We extensively evaluate the performance of *Harmony* on popular testbeds under artificial RF interference, showing significant performance improvements over state-of-the-art CT-based protocols (Sect. IV).

After describing related work in Sect. V, we conclude our paper in Sect. VI, along with a discussion on future work.

II. Towards a Dependable Data Collection in Harsh RF Environments

As discussed in Sect. I, the presence of radio interference in the surroundings of a network undermines the dependability of CT schemes, as they (i) rely on a tight, μs-level synchronization, and (ii) incur a significant energy overhead to counteract packet loss. Because of the increasing congestion of the ISM bands, we seek to design a protocol that fully exploits the potential offered by CT, while being able to withstand harsh interference with reasonably balanced performance over time.

We discuss next the challenges in sustaining such a dependable and balanced performance under harsh RF interference. We further highlight which mechanisms should be embedded in the design of an interference-resilient CT-based protocol.

Using multiple channels. When it comes to interference-resilient CT-based solutions, the community has mostly focused on the use of *channel hopping* techniques [18], [19], [26]. Some solutions embed the use of multiple channels within multiple CT-slots [26], whereas others make use of the same channel during a flooding round and change channel across subsequent floods [19][1]. Whilst both approaches explore frequency diversity, they do not allow to deliver a packet if interference persists on a given channel for an extended time. Indeed, if no potential recipient receives the packet, the information does not propagate and the flooding round ends.

Approaches making use of back-to-back transmissions [18] mitigate this problem, but require nodes to keep sending information for several CT-slots, and to remain accurately synchronized with the rest of the network to channel-hop (which is intrinsically hard and highly inefficient under interference).

To avoid the need of hopping synchronously and to minimize energy consumption, *Harmony* uses a *scan-and-lock* mechanism, where information is sent back-to-back over several frequencies, but where each node listens to only one channel rather than actively hopping every CT-slot. In particular, each node scans all available channels and locks to the least noisy one, as detailed in Sect. III-B. By doing so, the *scan-and-lock* mechanism avoids that nodes listen to congested channels and maximizes the chances that a node receives a message without a tight synchronization to the host, as packets are sent back-to-back on multiple frequencies.

Correctly defining node behaviour. In order to efficiently orchestrate the data collection in the network, nodes need to be aware of which packets have been correctly received by the sink and of which source nodes have data to send. This way, when all packets have been delivered correctly, nodes can turn off their radio and save energy. Similarly, as soon as a source node has a new packet to send, all relay nodes as well as the sink should wake up and be ready to receive or forward data. If several source nodes need to send a packet at the same time, they should do so using different slots to avoid collisions.

To this end, protocols like Crystal [10] make use of flooding rounds to acknowledge the reception of messages and inform source nodes whether re-transmissions are needed. However, if relay nodes do not receive packets due to the presence of interference, they assume no data needs to be forwarded and turn off their radio, which leads to an erroneous behaviour of the nodes in the network, increases end-to-end delays, and even incurs a packet loss.

In order to ensure a robust negotiation even under heavy interference, *Harmony* makes use of *negotiation-and-action* phases where all nodes reliably share their current status and determine their next actions, as described in Sect. III-A. Instead of considering the absence of packets as the condition to sleep as in [10], *Harmony*'s negotiation is *explicit*, meaning that a node turns off its radio only after successfully receiving a complete picture of the status of other nodes in the network.

[1]We refer to *CT-slot* as the concurrent transmission of a message to multiple receivers using Glossy [7]. A *flooding round* is composed of several consecutive CT-slots used to propagate (flood) information across the network.

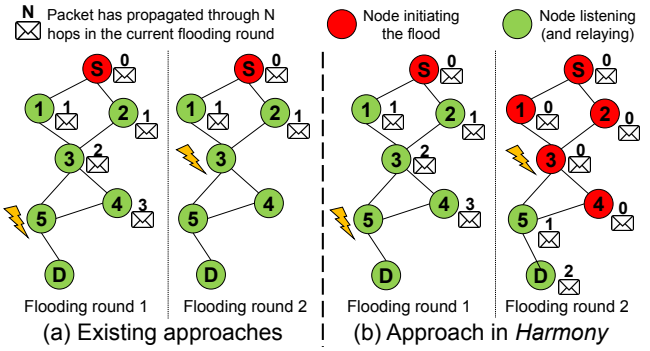

Fig. 1: Interference may partition the network, preventing the completion of a flood and leading to de-synchronization (a). In *Harmony*, additional nodes are allowed to transmit messages to maintain synchronization or to resume a previous flood (b).

Dealing with network partitions. The presence of persistent interference may be deleterious for the performance of CT-based protocols, as it may lead to network partitions. Especially the presence of wideband interferers can be critical: for example, when two or three co-located Wi-Fi devices are operating on orthogonal frequencies, all IEEE 802.15.4 channels can experience a significant packet loss [23]. This makes channel hopping mechanisms ineffective and makes it hard to (i) deliver messages correctly and (ii) keep nodes synchronized. When nodes lose synchronization to the host, they are unable to wake up synchronously and receive packets, which significantly degrades the network performance.

Fig. 1(a) exemplifies the problem by depicting a network in which a source node S (also acting as the host) attempts to initiate a flood towards a sink node D. If a long burst of interference persistently affects the reception on node 5, the information cannot be propagated to the sink by the end of the first flooding round. In the following round (which may make use of a different channel), the information gets only propagated until nodes 1 and 2, as this time node 3 is affected by harsh RF interference. As a result, node 5 and the sink D do not receive a message in either rounds: if this situation persists, the nodes may lose synchronization from the host and the information may remain undelivered for a long time.

Unfortunately, the scenario shown in Fig. 1(a) is rather common in existing CT-based solutions. An interference-resilient protocol, instead, would behave as depicted in Fig. 1(b). It allows one or more nodes (nodes 1 to 4 in this case) to initiate the flood besides S and/or act as reference clocks temporarily [20]. By doing so, the information gets propagated to the sink node despite the interference affecting node 3, and node 5 as well as D can remain synchronized. *Harmony* achieves exactly this by making use of an *agent-initiated negotiation* (seeing Sect. III-A) and a *data freezing* (seeing

Sect. III-B) technique that resumes flooding and improves reliability when the network is partitioned.

Avoiding long packets. The probability that a packet is hit by interference increases with the length of the message, as this is proportional to the time necessary to transmit a packet over the air [23]. For example, transmitting an IEEE 802.15.4-compliant packet with a 64-Byte payload requires 2.3 ms, whereas a packet with 8-Byte payload only takes 512 μs.

In principle, one could learn the characteristics of interference and wait for a sufficiently long white space to transmit information [27], [28]. In congested channels and in the presence of highly-dynamic interference, however, it may be hard to find and align to a sufficiently long white space.

Fragmenting larger packets into smaller chunks allows to maximize the probability of successful reception, at the price of an additional overhead. In *Harmony*, inspired by recent work making use of network coding to increase the effective capacity of a CT-based network [14]–[17], we divide longer packets into blocks and send a random combination of these blocks using LT codes [29]. Besides avoiding the transmission of longer packets, the use of coded packet in *Harmony* also helps to increase the chances of successful delivery over unreliable channels, as described in Sect. III-B.

Detecting interference in the network. The ability to detect the presence (and quantify the severity) of interference on a network is a fundamental stepping stone for the selection of an appropriate mitigation strategy at runtime [30].

$\text{Crystal}^{\text{CH}}_{\text{ND}}$ [19] is one of the few CT-based protocols that adaptively changes its behaviour at runtime when detecting abnormally high interference. As mentioned previously, the basic Crystal protocol regards the absence of packets as a sign to turn off the radio and enter sleep mode. This is often a problem in the presence of interference, as source nodes may actually be trying to send data that is lost while being flooded through the network. To mitigate the problem, $\text{Crystal}^{\text{CH}}_{\text{ND}}$ extends Crystal by letting each node perform a clear channel assessment (CCA) periodically and keep the radio active when significant noise is detected (i.e., when a large number of CCA checks detect a busy channel). This way, as soon as interference clears, nodes are still active and can forward the pending messages towards the intended destination.

This approach is not ideal, as it assumes that the noise measured by a node is representative of the one experienced by other nodes in the network. In real-world scenarios, however, the interference distribution varies across space and may only affect a fraction of the nodes. As a result, a node may remain awake because it measures a high interference in its surroundings, even though no message was actually transmitted, which results in an unnecessary energy waste.

A much better approach would inform a node about the presence and distribution of interference across the *whole* network, such that the node can autonomously decide the corresponding interference mitigation technique. For example, in the network shown in Fig. 1, if node S needs to send a message to D and is informed that there may be a strong interference nearby nodes 3 and 5, it can use network coding to fragment information or send the same message twice to maximize the chances of a successful and timely delivery.

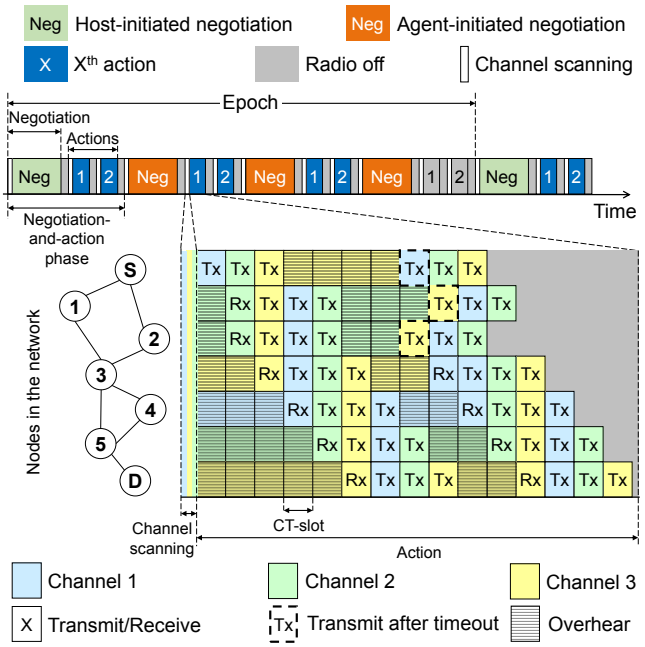

Fig. 2: Overview of *Harmony*'s design: time is split in epochs, during which several negotiation-and-action phases take place.

In *Harmony*, we design a mechanism called *SNI* that allows each node to build a bitmap capturing the interference distribution across the network. This way, a node can grasp the severity of the congestion and use more (or less) aggressive schemes to deliver packets. As shown in Sect. III-C, the SNI allows nodes to autonomously decide which interference mitigation features to enable and achieve a balanced performance.

III. *HARMONY*: Design and Implementation

We describe next the design and implementation of *Harmony* in detail. We first provide an overview of the protocol and its basic components (Sect. III-A). We then present several mechanisms that make *Harmony*'s transmissions robust to harsh RF interference (Sect. III-B) and describe how to activate them in order to achieve a good balance between reliability, timeliness, and energy efficiency (Sect. III-C). We finally illustrate how we implemented *Harmony* on top of the Contiki OS on a popular IEEE 802.15.4 platform (Sect. III-D).

A. Protocol Overview and Basic Components

Harmony is a CT-based data collection protocol in which the nodes in the network can take over three roles: *source* (generating messages), *relay* (forwarding messages), and *sink* (collecting messages). In our implementation we assume that there is only one sink in the network. One of the nodes in the network also acts as a *host*, i.e., as reference clock.

Fig. 2 sketches the high-level design of *Harmony*: time is split in *epochs*, during which a number γ of *negotiation-and-action* phases take place ($\gamma = 4$ in this illustration).

Negotiation-and-action phase. Each of these phases consists of one *negotiation* and multiple *actions*. A negotiation consists

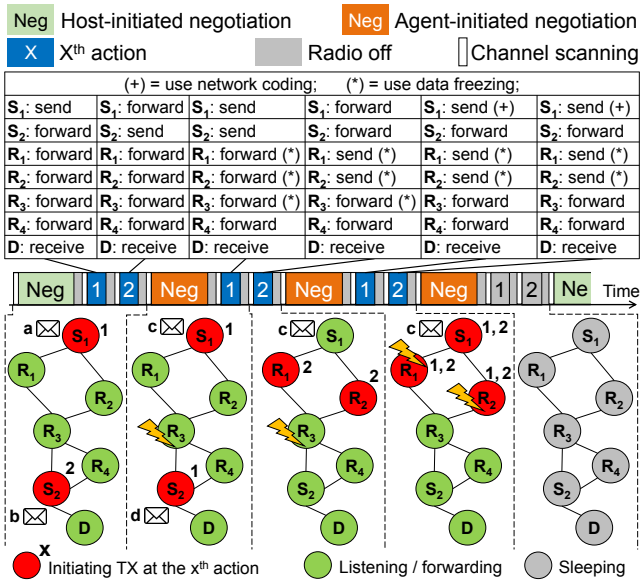

Fig. 3: The agent-initiated negotiation, together with the use of network coding and data freezing techniques, allows *Harmony* to reliably deliver packets despite network partitions.

of a many-to-all/all-to-all data sharing session during which the source nodes indicate whether they have data to send and the sink node acknowledges the successful reception of packets. Source nodes also share whether they have nothing to send, i.e., in normal conditions, nodes always receive information from every source node in the network. Note that the all-to-all data sharing utilizes CT and in-network processing together [8], completed within 80 ms in our implementation.

Therefore, after a negotiation, all the nodes in the network know how they should act, i.e., whether they should remain awake to forward packets, (re-)transmit information, or turn off their radio for the following θ actions ($\theta = 2$ in Fig. 2) of a pre-defined duration. Each action is a CT flooding round, long enough to forward information from one source to the sink by using CT flooding[2]. In case no source node has data to send, all nodes enter low-power mode until the next negotiation.

Agent-initiated negotiation. *Harmony* supports two types of negotiation: *host-initiated* and *agent-initiated*, marked in Fig. 2 as green and orange, respectively. During the agent-initiated negotiation, one or more synchronization agents are allowed to initiate a flood besides the host and pretend to be reference clocks. As highlighted in Sect. II, this allows nodes in the network to maintain (local) synchronization to at least one of the synchronization agents despite network partitions.

Fig. 3 illustrates this in more detail by showing the exemplary behaviour of a network with two source nodes (S_1 and S_2, with the former acting as a host), a sink node D, and four relay nodes (R_1 to R_4) during an epoch with $\gamma = 4$ negotiation-and-action phases. Both S_1 and S_2 have two packets to send during this epoch. The first negotiation is initiated by the host (S_1): as the network is interference-free, at the end of the negotiation all nodes are aware that S_1 and S_2 have data to send. Hence they will behave accordingly during the following two actions: S_1 will send packet a during the first action, whereas S_2 will transmit packet b during the second one; all other nodes will forward data to D.

The second negotiation is agent-initiated, i.e., it is initiated by the host as well as several other synchronization agents (R_2 and S_2 in this example). During this negotiation, a strong interference is present in the surroundings of node R_3, blocking all its communications. Even though interference is partitioning the network, the nodes can maintain synchronization thanks to the synchronization agents and can decide locally on a series of actions. In particular, node S_2 can schedule the transmission of packet d to the sink D, which receives the packet successfully.

Network-wide interference estimation. After every negotiation (both host-initiated and agent-initiated), the nodes derive information about the presence of interference throughout the network. As every source node and the sink share their status, all nodes can infer the presence of interference through the network by analysing whether some of the messages from these nodes were not received during the negotiation.

This interference detection mechanism, named *SNI* (sensing network-level interference) and detailed in Sect. III-C, is used by every node to decide whether additional features should be activated in an attempt to sustain a more reliable delivery under interference. Such features, described in Sect. III-B, include the transmission of coded packets and a data freezing mechanism resuming flooding in a partitioned network.

In the example shown in Fig. 3, after the second negotiation, S_1, R_1, and R_2 can derive that the network is partitioned, as they do not receive any information from S_2 and D; R_3 does not receive any packet due to the interference and infers it is isolated from the network; nodes S_2 and R_4 infer that the network is partitioned, but that they have a reliable path to D.

B. Ensuring Reliable Delivery in Harsh RF Environments

Based on the SNI, nodes can decide whether activating some of the features embedded in *Harmony* that help increasing the reliability under harsh RF interference.

Data freezing. In Fig. 3, after the second negotiation, nodes R_1, R_2, and R_3 know that there is interference on the path towards D and that S_1 has data to send. They can hence decide to enable *Harmony*'s *data freezing* (DF) mechanism, during which relay nodes can initiate an action with the packet that was received during previous actions. In this example, during the first action following the second negotiation, node S_1 transmits packet c towards D. Because of the persistent interference on R_3, this message will only be received by R_1 and R_2. These two nodes, who activated DF, will store c and attempt its transmission during the next actions, unless the sink acknowledges its reception.

During the third negotiation, a strong interference is affecting nodes R_1 and R_2, which do not receive messages from other nodes and infer that strong interference is present nearby. Therefore, they keep the DF mechanism activated and initiate

[2]The action in which a source node transmits information changes after every negotiation and can be computed locally based on the number of actions θ and the number of the current negotiation, as detailed in Sect. III-D.

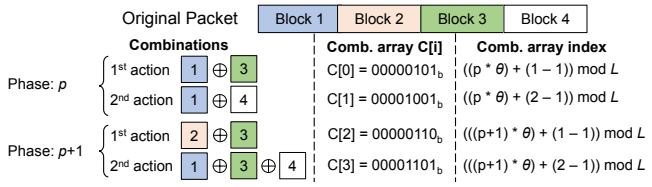

Fig. 4: *Harmony* can use LT codes to shorten the packet length and increase the robustness of transmissions to interference.

the flood of packet c in the following two actions, during which the message will successfully reach D. As *Harmony* appends a sequence number to all outgoing messages (see Sect. III-D), the sink is able to filter out the duplicated message.

Nodes using DF store a packet for at most p_{df} negotiation-and-action phases, with p_{df} being a parameter configurable at compile time ($p_{df} = 2$ in our implementation). This way, all nodes using DF will forget a packet simultaneously.

Network coding. In *Harmony*, any node can use Luby Transform (LT) codes [29] to shorten the packet length and increase the robustness of transmissions to interference. Specifically, a packet P with a length of l Bytes is divided into B blocks of $\lceil l/B \rceil$ Bytes each, i.e., $P = \{x_1, x_2, x_3, ..., x_B\}$. A different combination of these blocks is sent by a node in successive actions based on a pre-defined combination array C of length L, as illustrated in Fig. 4.

To reduce the overhead, *Harmony* does not add inside every packet the combination array embedding the information of which blocks are being carried. Instead, it computes this information from the sequence number of the current negotiation-and-action phase p, the number of actions in each phase θ, and the current action's number, as shown in Fig. 4. The use of agent-based negotiation ensures that nodes always know p (even when a network is partitioned) and decode the original packet without the need of extra information. In the Fig. 3 example, after the third negotiation, S_1 notices that c was not acknowledged by D for the second time in a row, and hence decides to use network coding during the following actions.

Scan-and-lock. To make an efficient use of channel hopping, *Harmony* adopts a *scan-and-lock* mechanism, as discussed in Sect. II. In this scheme, inspired from [20], each node scans the available channels before each negotiation/action, and locks to the least noisy one, as illustrated in the bottom of Fig. 2. A node first scans the received signal strength (RSS) over the set of channels used (three in this example). It then picks the least noisy one (i.e., the one with the lowest RSS) and listens to only this frequency. When a node needs to initiate the flood or to relay a message, it transmits packets back-to-back across all given channels: this way all other nodes can receive the packets opportunistically. To maximize the reception of packets under interference, nodes can trigger a re-transmissions in case no data has been received within a given number of CT-slots.

C. Ensuring Balanced Performance

The data freezing and the network coding features are only effective under interference. If the network is interference-free, these approaches may increase energy consumption, communication latency, or even reduce the reliability. To activate them only when strictly necessary, *Harmony* makes use of the SNI mechanism to get a picture of the connectivity in the network.

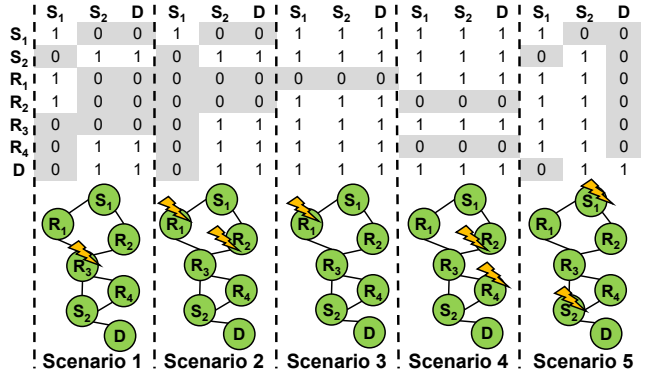

Fig. 5: *Harmony*'s SNI mechanism allows nodes to identify the presence and severity of interference throughout a network.

The SNI mechanism works as follows: at the end of every negotiation, nodes compute a bitmap similar to the one shown in Fig. 5. The bitmap is built by gathering the information collected during the negotiation from every source node in the network as well as the sink. As discussed in Sect. III-D, the packets exchanged during the negotiation contain a field where each non-relay node adds information. The bitmap is built by marking with "1" each non-relay node whose information has been added to this packet. Thus, a node is able to infer the connectivity with the sink and with other source nodes, as well as the severity of network partitioning.

Fig. 5 illustrates how the bitmap is built by every node in the network depicted at the bottom. Scenarios 1 and 2 correspond to the second and third negotiations described in Fig. 3, respectively. In scenario 1, nodes S_1, R_1, R_2, and R_3 realize about the network partition, whereas in scenario 2, node R_3 realizes that it has a connection to the sink (but not to S_1).

Scenarios 3 and 4 show how the SNI mechanism allows to also understand the *severity* of the interference in the network. In these scenarios, some of the relay nodes are affected by interference, but there are alternative paths allowing the sink and each source node to exchange information. Thanks to the SNI, these nodes know that they do not need to enable DF or network coding, hence preserving their energy budget.

By exploiting the SNI information after every negotiation, *Harmony* enables and disables interference mitigation strategies only when strictly necessary, achieving a balanced performance between reliability, timeliness, and energy efficiency, as shown by the evaluation results presented in Sect. IV.

D. Implementation

We implement *Harmony* using the popular Contiki operating system [24], on top of the off-the-shelf TelosB platform [31].

The implementation consists of two modules: a network and an application module. The *network module* embeds the primitives used for the actions (flooding based on the scan-and-lock mechanism) and for the negotiations (many-to-all/all-to-all communication). These primitives are called by a process running in the *application module*. The latter is in charge of making decisions (e.g., enter low-power mode, remain active

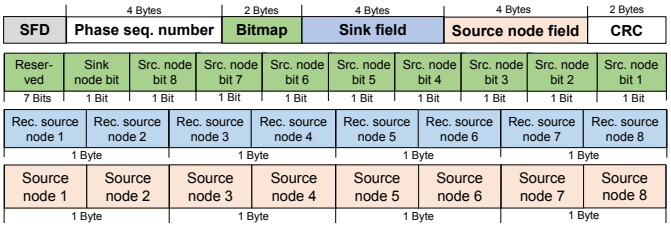

Fig. 6: Format of the packet used in *Harmony*'s negotiation, assuming a network with $\phi = 8$ source nodes and one sink.

to forward packets, enable an interference mitigation strategy) based on the bitmap returned by the SNI after a negotiation.

During a negotiation, all nodes exchange packets with the format shown in Fig. 6. The sink node sets to 1 the corresponding bit in the bitmap field and enters the last 4 bits of the sequence number of the last received packet from each source node. Every source node sets to 1 the corresponding bit in the bitmap field regardless of whether it has data to transmit. If this is the case, the source node enters also the last 4 bits of the sequence number of the packet to be sent.

Each negotiation-and-action phase is assigned a 4-Byte sequence number that is entered in every packet sent during the negotiation. The sequence number of the current phase p is used to determine the order in which source nodes having packets to transmit initiate the flood. Specifically, given M source nodes $\{S_0, S_1, ..., S_{M-1}\}$ in a network, out of which N of them $\{s_0, s_1, ..., s_{N-1}\}$ have data to send in the current action, the node whose index is $s_{(p \times \theta + f) \ mod \ N}$, with f being the current action, initiates the flood; whereas all the other source nodes in the network act as relay. This guarantees that only one source node sends a packet at a given time when the negotiation is successful. Note that the packet format shown in Fig. 6 contains only up to $\phi = 8$ source nodes. If the total number of source nodes in the network M is higher than ϕ, a source nodes S_i participates actively only into the k^{th} negotiation, where $k = (i/\phi) + 1$, so to avoid collisions.

If a negotiation is unsuccessful due to interference (i.e., at least one source node or the sink has a bitmap bit set to 0), in the following θ actions all the relay nodes and the sink keep their radio on. The source nodes, instead, will send a packet with a given probability (set to 50% in our current implementation) or act as a relay otherwise. This leaves some of the actions available to other nodes to send data. For example, in the scenario 5 of Fig. 5, both source nodes (S_1 and S_2) are interfered. Therefore, during the first action, S_2 acts as a relay, whereas S_1 sends its message, which reaches the four relay nodes (who have activated the DF mechanism, as they have no path to the sink D). During the second action, S_2 transmits its message, which is correctly received by D. If interference on S_2 clears in any of the following actions, D can get the original message sent by S_1 during the first action (either from S_1 directly or thanks to the DF mechanism). This probabilistic scheme hence helps to increase the chance of successful packet reception with unsuccessful negotiations.

A correct parametrization of *Harmony* is very important. The number of negotiation-and-action phases γ during an epoch directly affects the synchronization accuracy and should be increased if the network typically operates under strong interference. The number of actions θ in a negotiation-and-action phase should be low to minimize latencies, so that negotiations occur more frequently and nodes can signal that they have data to transmit. If low latencies are not necessary, this number can be increased to avoid frequent energy-expensive negotiations.

IV. Evaluation

We evaluate the performance of *Harmony* experimentally using two popular large-scale testbeds: FlockLab [32] and D-Cube [33]. Our evaluation answers the following questions:

- How does *Harmony* perform under interference? Do the DF and network coding features really help increasing the reliability under harsh interference? (Sect. IV-A)
- How does *Harmony*'s performance compare to that of state-of-the-art CT-based protocols? (Sect. IV-B)

Testbeds. We run our tests on *FlockLab* [32] and *D-Cube* [33], which embed 27 and 51 TelosB nodes, respectively. We select FlockLab, as it is one of the reference testbeds used by the community to evaluate the performance of low-power wireless protocols. We also make use of D-Cube, as it allows to accurately profile latency and power consumption in hardware, as well as to generate repeatable Wi-Fi interference [23].

Metrics. We compute three metrics during our experimental campaign: (i) average end-to-end *reliability*, average end-to-end *latency*, and average *power consumption*. The average end-to-end reliability is the ratio between the number of received and sent packets by all source nodes. The end-to-end latency is computed on all received packets as the time elapsed between the generation of a packet by a source node and the time in which it is received by the sink. The power consumption is the average across all the nodes in the network.

On the D-Cube testbed, a specific GPIO pin of the TelosB node is used to indicate whether a packet has been generated or received: the end-to-end latency is hence the difference between the timestamps of two GPIO events [34]. On FlockLab, application packets are generated by source nodes and timestamped using serial output logs. The average power is computed in hardware on D-Cube, and in software [35] as the percentage of time in which the radio is active on FlockLab.

RF interference. To test the performance of *Harmony* and other protocols under interference, we use JamLab [25] and JamLab-NG [23] to generate artificial noise in a repeatable way. In FlockLab, we let a fraction of the nodes in the testbed (nodes 33, 32, 28, 10, 19, and 13)[3] run JamLab on IEEE 802.15.4 channel 26 using the highest transmission power available (0 dBm). On D-Cube, we let each Raspberry Pi 3 node (connected to the TelosB node) generate wideband Wi-Fi interference on a random Wi-Fi channel using JamLab-NG [23].

We distinguish between two types of interference: *mild* and *strong*. In D-Cube, mild and strong interference are generated by running `Confiture` over slots of 13 ms with a channel

[3]The topology of FlockLab is available at https://user.flocklab.ethz.ch/user/topology.php

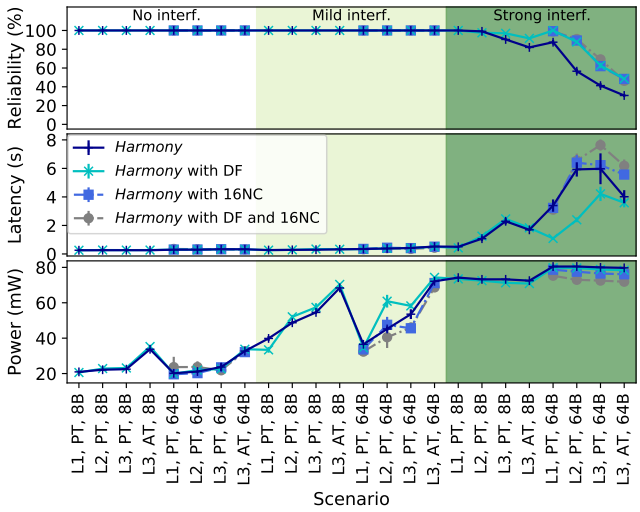

Fig. 7: Performance of *Harmony* in D-Cube with and without enabling the data freezing and network coding features.

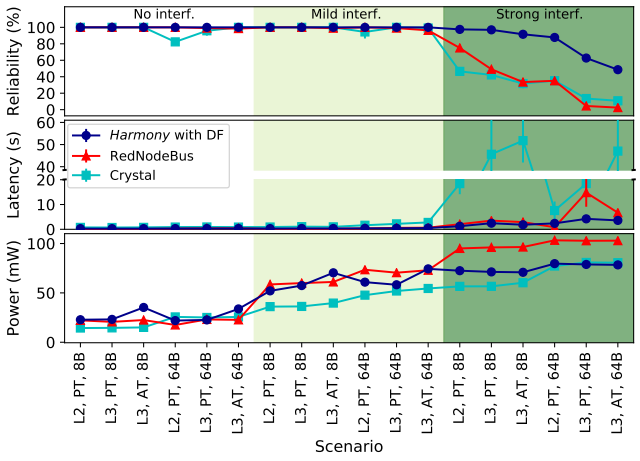

Fig. 8: In D-Cube, *Harmony* outperforms state-of-the-art CT-based protocols under strong interference in terms of reliability and timeliness, while being comparably energy-efficient.

occupancy of approximately 40 and 80%, respectively. Strong interference makes use of 11 dB higher transmission power than the mild one. In FlockLab, mild and strong interference are generated by dividing time in slots of 4 ms, and by permanently jamming a slot with a chance of 50 and 70%, respectively. Each experiment is also run in absence of interference, so to have a reference performance in ideal settings.

Scenarios. In each testbed we use multiple many-to-one layouts, i.e., diverse configurations of source nodes and sink, to make sure that our results are general and not setup-specific. In FlockLab, we select:

- Layout 1: $\{4, 7, 8, 11, 14, 16, 17, 24\} \rightarrow 1$
- Layout 2: $\{8, 11, 17\} \rightarrow 16$

In D-Cube, we select:

- Layout 1: $\{119, 207, 212, 220, 222\} \rightarrow 202$
- Layout 2: $\{111, 203, 210, 217, 219, 225\} \rightarrow 208$
- Layout 3: $\{112, 200, 202, 205, 209, 220, 224, 225\} \rightarrow 226$

The maximal distance between a source node to the sink is 4 hops in FlockLab and 6 hops in D-Cube. For each of these layouts (denoted as L1–L3), we test the protocol performance using different packet lengths. In particular, we use an application payload of 8 and 64 Bytes (denoted as 8B and 64B, respectively). We also test the impact of traffic load by making use of periodic traffic (PT) and aperiodic or asynchronous traffic (AT). In PT, each node generates a packet every 30 s on both D-Cube and FlockLab. In AT, packets are generated at random intervals (D-Cube) or every 30 s, but with different offsets across nodes (FlockLab). Each experiment runs for 10 minutes, and is repeated at least three times.

A. Performance of Harmony under interference

We first analyse the performance of *Harmony* with different configurations on D-Cube. Fig. 7 shows the reliability, end-to-end latency and power consumption of *Harmony* with and without the data freezing and network coding features enabled.

First, we can notice that the basic configuration of *Harmony* sustains a near-perfect reliability in all testbed layouts even in the presence of mild interference (which causes a channel occupancy of 40% on several Wi-Fi channels). This confirms the goodness of the *scan-and-lock* mechanism, as well as of the *negotiation-and-action* and *agent-initiated negotiation*, which help sustaining a reliable performance. Under strong interference, the performance of *Harmony*'s basic configuration decreases especially when making use of large packets (64B).

We can also see that data freezing and network coding effectively help increasing the reliability under strong Wi-Fi interference by up to 30%. As expected, the use of data freezing helps reducing the end-to-end latency. The use of network coding, instead, slightly increases the end-to-end latency (as more messages need to be received and reassembled), but allows to reduce the energy consumption (as packets are smaller and the radio-on time can be kept shorter). A combined use of network coding and data freezing allows to reduce the energy consumption, but results in higher latencies.

B. Comparison to the state-of-the-art

We compare next the performance of *Harmony* to state-of-the-art CT-based solutions such as $\text{Crystal}_{\text{ND}}^{\text{CH}}$ [19] and RedNodeBus [12]. We do so by examining the performance of these three protocols during a public hackathon[4] [36], where the protocol developers have run their solution under the same settings for several months on the D-Cube testbed.

Fig. 8 shows the performance of the aforementioned protocols for different configurations. Overall, all protocols perform very reliably in absence of interference and in the presence of mild interference, with $\text{Crystal}_{\text{ND}}^{\text{CH}}$ showing a remarkable energy efficiency. However, under strong interference, *Harmony* significantly outperforms both $\text{Crystal}_{\text{ND}}^{\text{CH}}$ and RedNodeBus in terms of reliability and latency, while sustaining a comparable power consumption. In particular, *Harmony* achieves a 100% reliability in all scenarios involving mild interference, and up to 50% higher reliability under strong interference.

As Mixer [17] did not take part in the hackathon, we compare its performance to the one of *Harmony* by running

[4]We participated to the EWSN 2019 dependability competition as Team 01.

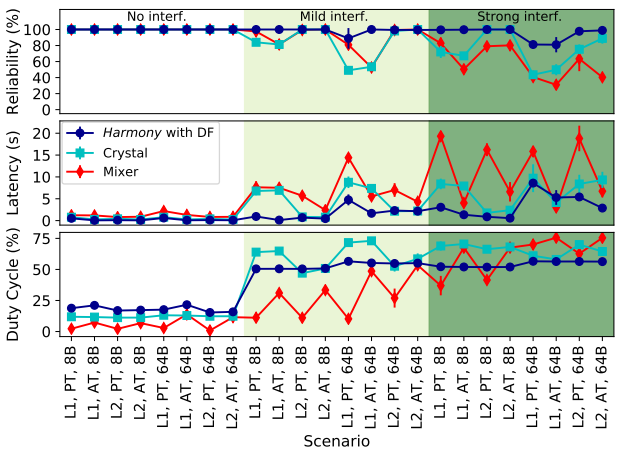

Fig. 9: Also in FlockLab, *Harmony* outperforms state-of-the-art protocols under strong interference in terms of reliability and timeliness, while being comparably energy-efficient.

tests in FlockLab with configurations similar to those used in D-Cube. We run Mixer[5] with node 4 and 8 as host in L1 and L2 (same setting used by *Harmony*), and use as Mixer's round period 30 s and 2.5 s for PT and AT, respectively, which allows to achieve a low latency for asynchronous traffic. In parallel to *Harmony* and Mixer, we also run $\text{Crystal}_{\text{ND}}$[6]. As Mixer does not embed channel hopping mechanisms and as JamLab can only interfere on channel 26, we disable channel hopping in *Harmony* and Crystal, so to ensure a more fair comparison.

Fig. 9 shows the performance of the three protocols for different configurations. The results show a similar trend to the ones in Fig. 8: *Harmony* exhibits the highest reliability and lowest latency under mild and strong interference. Mixer and Crystal show a lower energy consumption with no interference, but a comparable or higher power draw under strong interference. Hence, these experiments confirm the good balance that *Harmony* can achieve under harsh RF interference between reliability, end-to-end latency, and energy consumption.

V. Related Work

The pioneering work by Ferrari et al. [7], who led to the development of Glossy – a fast and efficient network flooding service by using CT – has revolutionized the design of low-power wireless protocols. By exploiting constructive interference and the capture effect on the MAC layer, Glossy provides highly reliable flooding for one-to-many communication. To realize data collection (many-to-one communication) with Glossy, Ferrari et al. [9] have added an application-level scheduler to construct a so-called low-power wireless bus (LWB). The latter centrally schedules the data communication to support one-to-many, many-to-one, and many-to-many communications. Crystal [10] focuses on data collection (many-to-one) applications and makes scheduling simpler than LWB by using transmission-acknowledgement slot pairs.

Splash [14] builds a tree pipeline [37] by exploiting Glossy in order to improve channel utilization. Pando [38] has integrated fountain code with pipelines to overcome the long-tail problem of Splash [14]. Both works, as well as Ripple [39] are designed to deliver large amounts of data through the network, e.g., for reprogramming the nodes, rather than for *data collection*. Chaos [8] builds on Glossy to achieve fast all-to-all data sharing. Codecast [16] provides a more general approach to support many-to-many communication by introducing feedback-driven network coding. More recently, Mixer [17] has improved the effectiveness of network coding by adding recording capabilities at each node.

These and other works have shown that CT can be used as a basis to build dependable protocols achieving a high reliability and low delays. In particular, a few works, especially in the context of the EWSN dependability competition [33], have focused on introducing channel hopping techniques [11], [18], [20] and noise detection strategies [19] to improve the performance of CT-based solutions under interference. However, none of these works has focused on the ability to sustain a high performance even in *congested RF environments* where interference covers the entire IEEE 802.15.4 RF spectrum.

Harmony closes this gap and introduces generic techniques allowing CT-based communications to deal with partitioned networks and very adverse RF conditions. To this end, *Harmony* revises and incorporates channel hopping and network coding strategies, together with novel data freezing and negotiation-and-action schemes. *Harmony* further improves the local noise detection strategy introduced by $\text{Crystal}_{\text{ND}}^{\text{CH}}$ by introducing a distributed interference detection scheme able to quantify the interference on a network-level. To the best of our knowledge, *Harmony* is the first work tackling all these challenges and providing a highly robust CT-based solution to the low-power wireless community.

VI. Conclusions

We have presented *Harmony*, a CT-based data collection protocol designed to sustain a reliable performance even when all available IEEE 802.15.4 channels are highly congested. To do so, *Harmony* includes, among others features allowing to maintain synchronization and successfully deliver data in a partitioned network. Furthermore, a novel interference detection scheme allows *Harmony* to send packets aggressively only when strictly necessary. A thorough experimental evaluation shows that these techniques are highly effective: *Harmony* outperforms state-of-the-art protocols and achieves a good balance between reliability, timeliness, and energy-efficiency even under strong interference and when sending large packets.

Acknowledgment

We would like to thank the Computer Engineering Group at ETH Zürich for providing the FlockLab testbed, as well as the authors of the D-Cube testbed from TU Graz. We also thank Pei Tian for her contribution on experiments, Reto Da Forno, and Markus Schuß for their technical support.

[5]https://gitlab.com/nes-lab/mixer, hash 0b64204c.

[6]https://github.com/d3s-trento/crystal, hash 11ef31e. Note that in Crystal, the sink acts as a host node by default: we hence used node 1 and 16 for L1 and L2, respectively. During our experiments on FlockLab, we fine-tuned the length of `T`, so to deliver long application packets. We have also disabled Crystal's auto-reset function, which triggers if a node did not receive a sync for a long time: this allows us to fully observe Crystal's best-effort performance.

REFERENCES

[1] J. A. Stankovic, "Research directions for the Internet of Things," *Internet of Things Journal*, vol. 1, no. 1, Feb 2014.

[2] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Deploying wireless sensor networks with fault-tolerance for structural health monitoring," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 382–395, 2013.

[3] H.-S. Kim, H. Cho, M.-S. Lee, J. Paek, J. Ko, and S. Bahk, "Marketnet: An asymmetric transmission power-based wireless system for managing e-price tags in markets," in *Proceedings of the 13th International Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, 2015, pp. 281–294.

[4] E. A. Lee, "Cyber physical systems: Design challenges," in *Proceedings of the 11th International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, May 2008.

[5] F. Foukalas, P. Pop, F. Théoleyre, C. A. Boano, and C. Buratti, "Dependable wireless industrial iot networks: Recent advances and open challenges," in *Proceedings of the 24th IEEE European Test Symposium (ETS)*. IEEE, May 2019.

[6] T. Chang, T. Watteyne, X. Vilajosana, and P. H. Gomes, "Constructive Interference in 802.15.4: A Tutorial," *Communications Surveys Tutorials*, vol. 21, no. 1, pp. 217–237, Sep 2018.

[7] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2011, pp. 73–84.

[8] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2013, pp. 1:1–1:14.

[9] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2012, pp. 1–14.

[10] T. Istomin, A. L. Murphy, G. P. Picco, and U. Raza, "Data prediction + synchronous transmissions = ultra-low power wireless sensor networks," in *Proceedings of the 14th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2016, pp. 83–95.

[11] A. Escobar, C. Gonzalez, F. J. Cruz, J. Garcia-Jimenez, J. Klaue, and A. Corona, "RedFixHop: Efficient ultra-low-latency network floodings," in *Proceedings of the 13th International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, Jun. 2016.

[12] A. Escobar-Molero, J. Garcia-Jimenez, J. Klaue, F. Moreno-Cruz, B. Saez, F. J. Cruz, U. Ruiz, and A. Corona, "RedNodeBus: Stretching out the preamble," in *Proceedings of the 16th International Conference on Embedded Wireless Systems and Networks (EWSN)*, Feb. 2019.

[13] M. Suzuki, S. Ohara, K. Jinno, C. H. Liao, and H. Morikawa, "Wireless-transparent sensing," in *Proceedings of the 14th International Conference on Embedded Wireless Systems and Networks (EWSN)*, Feb. 2017.

[14] M. Doddavenkatappa, M. C. Chan, and B. Leong, "Splash: Fast data dissemination with constructive interference in wireless sensor networks," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2013, pp. 269–282.

[15] W. Du, J. C. Liando, H. Zhang, and M. Li, "Pando: Fountain-enabled fast data dissemination with constructive interference," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 820–833, 2016.

[16] M. Mohammad and M. C. Chan, "Codecast: Supporting data driven in-network processing for low-power wireless sensor networks," in *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2018, pp. 72–83.

[17] C. Herrmann, F. Mager, and M. Zimmerling, "Mixer: Efficient many-to-all broadcast in dynamic wireless mesh networks," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2018, pp. 145–158.

[18] R. Lim, R. Da Forno, F. Sutton, and L. Thiele, "Competition: Robust flooding using back-to-back synchronous transmissions with channel-hopping," in *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2017, pp. 270–271.

[19] T. Istomin, M. Trobinger, A. L. Murphy, and G. P. Picco, "Interference-resilient ultra-low power aperiodic data collection," in *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2018, pp. 84–95.

[20] X. Ma, P. Zhang, X. Li, W. Tang, J. Wei, and O. Theel, "DeCoT: A dependable concurrent transmission-based protocol for wireless sensor networks," *IEEE Access*, vol. 6, pp. 73 130–73 146, 2018.

[21] T. Schmid, "Time in wireless embedded systems," Ph.D. dissertation, University of California, Los Angeles, CA, USA, 2009.

[22] C. A. Boano and K. Römer, "External radio interference," in *Radio Link Quality Estimation in Low-Power Wireless Networks*. Springer, Jul. 2013, pp. 21–63.

[23] M. Schuß, C. A. Boano, M. Weber, M. Schulz, M. Hollick, and K. Römer, "JamLab-NG: Benchmarking low-power wireless protocols under controlable and repeatable Wi-Fi interference," in *Proceedings of the 16th International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2019, pp. 83–94.

[24] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - A lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 1st International Workshop on Embedded Networked Sensors (EmNetS)*, Nov. 2004, pp. 455–462.

[25] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. Zúñiga, "Jamlab: Augmenting sensornet testbeds with realistic and controlled interference generation," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2011, pp. 175–186.

[26] P. Sommer and Y. A. Pignolet, "Competition: Dependable network flooding using Glossy with channel-hopping," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2016, pp. 303–303.

[27] G. J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond co-existence: Exploiting WiFi white space for Zigbee performance assurance," in *Proceedings of the 18th International Conference on Network Protocols (ICNP)*. IEEE, Oct. 2010, pp. 305–314.

[28] K. R. Chowdhury and I. F. Akyildiz, "Interferer classification, channel selection and transmission adaptation for wireless sensor networks," in *Proceedings of the International Conference on Communications (ICC)*. IEEE, Jun. 2009, pp. 1–5.

[29] M. Luby, "LT codes," in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2002, pp. 271–280.

[30] F. Hermans, O. Rensfelt, T. Voigt, E. Ngai, L.-Å. Nordén, and P. Gunningberg, "SoNIC: Classifying interference in 802.15.4 sensor networks," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN)*. ACM, Apr. 2013, pp. 55–66.

[31] *Tmote Sky datasheet*, Moteiv Corporation. [Online]. Available: http://www.eecs.harvard.edu/ konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

[32] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, and J. Sommer, P.and Beutel, "FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proceedings of the 12th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2013, pp. 153–166.

[33] M. Schuß, C. A. Boano, M. Weber, and K. Römer, "A competition to push the dependability of low-power wireless protocols to the edge," in *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2017, pp. 54–65.

[34] M. Schuß, C. A. Boano, and K. Römer, "Moving beyond competitions: Extending D-Cube to seamlessly benchmark low-power wireless systems," in *Proceedings of the 2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, 2018, pp. 30–35.

[35] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He, "Software-based online energy estimation for sensor nodes," in *Proceedings of the 4th International Workshop on Embedded Networked Sensors (EmNetS)*, Jun. 2007, pp. 28–32.

[36] "EWSN 2019 Dependability Competition." [Online]. Available: https://iti-testbed.tugraz.at/blog/page/21/ewsn-19-dependability-competition-final-results/

[37] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale, "PIP: A connection-oriented, multi-hop, multi-channel TDMA-based MAC for high throughput bulk transfer," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2010, pp. 15–28.

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

[39] D. Yuan and M. Hollick, "Ripple: High-throughput, reliable and energy-efficient network flooding in wireless sensor networks," in *Proceedings of the 16th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015, pp. 1–9.