Less is More: Learning More with Concurrent Transmissions for Energy-Efficient Flooding

Peilin Zhang[†] Alex Yuan Gao[*] Oliver Theel[†]

[†] Department of Computer Science
Carl von Ossietzky University of Oldenburg, Germany
{peilin.zhang, theel}@informatik.uni-oldenburg.de

[*] Department of Information Technology
Uppsala University, Sweden
alex.yuan.gao@it.uu.se

ABSTRACT

Concurrent transmissions, a novel communication paradigm, has been shown to effectively accomplish a reliable and energy-efficient flooding in wireless networks. With multiple nodes exploiting a receive-and-forward scheme in the network, this technique inevitably introduces communication redundancy and consequently raises the energy consumption of the nodes. In this paper, we propose LiM, an energy-efficient flooding protocol for wireless sensor networks. LiM builds on concurrent transmissions, exploiting constructive interference and the capture effect to achieve high reliability and low latency. Moreover, LiM equips itself with a machine learning capability to progressively reduce redundancy while maintaining high reliability. As a result, LiM is able to significantly reduce the radio-on time and therefore the energy consumption. We compare LiM with our baseline protocol Glossy by extensive experiments in the 30-node testbed FlockLab. Experimental results show that LiM highly reduces the broadcast redundancy in flooding. It outperforms the baseline protocol in terms of radio-on time, while attaining a high reliability of over 99.50%, and an average end-to-end latency around 2 ms in all experimental scenarios.

CCS CONCEPTS

• **Networks → Network protocol design**;

KEYWORDS

Flooding, Data Dissemination, Machine Learning, Exp3, Wireless Sensor Networks

1 INTRODUCTION

Over the past decade, Wireless Sensor Networks (WSNs) began to play a significant role as an enabling technology in a large number of applications, including health-care, industry, agriculture, and transportation. Network flooding is a protocol that delivers messages from a source node to all other nodes in a connected network. As an essential operation for WSNs, flooding is widely used for information dissemination, code update, time synchronization, and network configuration, to name but a few. Recently concurrent transmission (CT)-based flooding has been introduced in low-power wireless networks as a promising technique. For example, Flash Flooding [1] and Glossy [2] exploit constructive interference and the capture effect to achieve highly reliable data flooding in multi-hop WSNs. These protocols significantly increase network throughput, enhance packet transmission reliability, and reduce flooding latency. However, these protocols have to introduce high communication redundancy in order to attain high reliability. That is, to obtain a fast and reliable coverage of the whole network, each sensor node has to broadcast the received packet until every node in the network has been covered. Consequently, there exists a large number of transmission redundancy, i.e., many of these broadcast transmissions are not necessary. In this case, sensor nodes consume much more energy than expected. This type of aggressive flooding, generally referred to as blind flooding [3], is not energy-efficient.

Moreover, CT-based flooding also suffers from a scalability problem with respect to the temporal misalignment among base-band signals. Namely, the packet reception rate degrades as the node density or the size of the network increases. As discussed by the authors in [4, 5], the probability of receiving a packet due to the capture effect drops notably, as the number of synchronous transmitters increases. To overcome these problems, Chaos [4] exploits in-network processing together with concurrent transmissions: While each node receives a packet, it spends a fixed period of time (processing time) to process the data, and then makes a decision whether it is necessary to forward the received packet. In this case, it is able to appropriately decrease the number of concurrent transmitters, and maintain the best-effort performance even in high-density WSNs.

Furthermore, CXFS [6] concentrates on one-to-one data transmission and it builds a forwarder selection scheme on CT-based flooding. CXFS aims to reduce the wasteful transmissions, thus to improve energy efficiency and throughput, while providing a similar reliability. While Chaos and CXFS are based on Glossy, they still belong to blind flooding after all. That means, to achieve a high reliability, these Glossy-based flooding protocols are required to repeat the transmission for a fixed number of times. For instance, the Glossy protocol (referred to as "Glossy" in the following context) sets the maximum number of transmission to 5 by default to accomplish high reliability.

In order to avoid blind flooding but at the same time maintain high reliability, each sensor node should be able to decide whether or not it is essential to forward the received packet, based on the current environmental conditions. Decisions are adaptively made

MobiQuitous 2017, November 7–10, 2017, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5368-7/17/11…$15.00
https://doi.org/10.1145/3144457.3144482

to maintain a good performance of the network, while minimizing transmission redundancy. Therefore, this decision-making adaptation can be converted to an optimization problem.

Generally, reinforcement learning techniques are effectively applied to solve these types of optimization problems. Reinforcement learning is a class of learning algorithms that attempts to maximize the cumulative reward by taking a specific action in a given state and following a predefined policy thereafter. Additionally, among all the reinforcement learning techniques, a set of so-called multi-armed bandit algorithms is particularly suitable for the optimization of the network. That is, the number of transmissions in each sensor node can be furthermore modeled as a multi-armed bandit problem, originally described by Robins [7]. A multi-armed bandit, also called K-armed bandit, is similar to a traditional slot machine but generally with more than one lever. This type of multi-armed bandit algorithm investigates the selection of the "best" action for the dynamic situations in low-power and lossy WSNs.

To this end, we propose LiM, a machine learning-based data dissemination protocol for low-power multi-hop WSNs. In designing LiM, we utilize a reinforcement learning technique to reduce redundant broadcast transmissions. We model the optimization of the transmission times in each sensor node as a multi-armed bandit problem. Besides, we exploit an exponential-weight algorithm for exploration and exploitation (called *Exp3*) for bandit learning in each sensor node of the WSN. Incorporated with concurrent transmissions, LiM is able to effectively achieve high end-to-end reliability and low end-to-end latency. Moreover, LiM empowers sensor nodes with a learning capability to reduce the redundancy of the flooding step by step, significantly lowering the power consumption. We implement LiM in Contiki [8] and conduct extensive experiments in a 30-node testbed – FlockLab [9]. Furthermore, we compare LiM to our baseline protocol Glossy [2] focusing on reducing communication redundancy in flooding.

Our evaluation shows that LiM is able to effectively limit the number of transmissions of the sensor nodes while still preserving high reliability and energy efficiency, as well as low latency: Sensor nodes, that do not belong to the backbone of the network, stay only in receiving and sleeping mode. The others execute the decision-making based on their obtained experience from the learning phase. With various levels of transmission power and different topologies, LiM obtains an average reliability of over 99.50%, and an average end-to-end latency of less than 2.5 ms in all experimental scenarios. Moreover, LiM reduces the radio-on time by at least 30% compared to the default configuration of Glossy.

We make the following contributions in this work:

- We propose LiM, an energy-efficient flooding protocol with progressive learning ability for low-power multi-hop WSNs.
- As a basis for LiM, we explore and implemented a light-weight bandit learning scheme to determine the number of broadcasts in each node. It guarantees a correct exploration of the "redundant" nodes and further conducts a progressive learning of the other nodes to greatly reduce broadcast redundancy.
- We implemented LiM in Contiki OS and conducted extensive experiments with various configurations in a 30-node real-world testbed. After that, we evaluated the performance of LiM in terms of end-to-end reliability, radio-on time, and latency.

The remainder of this paper is organized as follows. Section 2 explains the basis of LiM and provides a brief overview. Section 3 details the design perspectives of LiM, followed by performance evaluations elaborated in Section 4. Section 5 discusses related work, with two foci: on Glossy-based flooding protocols and on bandit learning strategies applied in WSNs. Section 6 concludes our work and leads to an outlook for future work.

2 OVERVIEW

In this section, we explain the basis of LiM in two dimensions: i) reliable flooding and ii) machine leaning. Then, we provide a brief overview of the protocol.

2.1 Reliable Flooding

Proposed in 2011, Glossy [2] is one of the most representative CT-based flooding protocols in the community. Basically, Glossy exploits both, constructive interference to avoid the contention of the communication medium access, and the capture effect to ensure that a receiver is able to correctly demodulate a received packet. These two mechanisms are able to empower Glossy to manage a highly reliable flooding and an accurate time synchronization.

2.1.1 Constructive Interference. Constructive interference occurs only when two or more nodes transmit *identical* packets. With IEEE 802.15.4 radios operating in 2.4 GHz ISM band, these identical packets are required to overlap within 0.5 μs [2, 10], that makes the signals appropriately superposed. Glossy manages this tight bound by using a radio-triggered synchronization mechanism.

2.1.2 Capture Effect. The capture effect is a phenomenon, where the receiver can lock on to and correctly demodulate the signal, when a received signal is approximately 3 – 4 dB stronger than the sum of all the other received signals [2, 11]. Besides, in IEEE 802.15.4 wireless networks, the strongest signal must arrive no later than 160 μs after the weaker signals [4], in order to be properly recognized and decoded by the receiver.

2.2 Machine Learning

Machine learning is a sub-field of artificial intelligence that is concerned with the question of "how to construct computer programs that automatically improve from experience" [12]. This property makes the family of machine learning-based algorithms being attractive for reliable and efficient communications in WSNs.

2.2.1 Reinforcement Learning. Reinforcement learning is one of the machine learning techniques, in which the learning agent earns the knowledge from the interaction with the environment. Technically, reinforcement learning is a class of algorithms that seeks to maximize the cumulative reward by executing different actions in a task. In this case, different configurations of a system can be modeled as the corresponding actions to maximize the reward in order to optimize the performance of the system.

2.2.2 Multi-armed bandit problem. The well-studied multi-armed bandit problem was originally proposed by Robbins [7] in 1985. A gambler firstly chooses K slot machines to play. At each time step, the gambler pulls one arm of one machine (out of K), and then

receives a positive, zero or negative reward. The purpose is to maximize the total reward over a sequence of trials. Assuming each arm in a slot machine has a different distribution of rewards, the goal is to find out the arm with the best expected return as early as possible.The problem is a classical example of the trade-off between exploration and exploitation [13]: On the one hand, if the gambler plays exclusively on the machine which the gambler supposes to be the best one ("exploitation"), then the gambler may fail to discover that one of the other arms in fact has a higher average return. On the other hand, if the gambler spends too much time trying out all K machines and then makes a decision based on the gathered statistics ("exploration"), then the gambler may fail to play the best arm for a long enough period of time to get a high total return.

To solve the multi-armed bandit problem, the exponential-weight algorithm for exploration and exploitation (Exp3) was proposed in 2002 by Auer et al. [14]. Exp3 is based on a reinforcement learning scheme and it solves the following problem: "if there are many available actions with uncertain outcomes in a system, how should the system act to maximize the quality of the results over many trials ?" We provide the details of Exp3 and the related implementation issues later in Section 3.

2.3 LiM in a Nutshell

LiM exploits both, constructive interference and the capture effect to guarantee a good performance of the network. However, the packet in LiM is not necessarily identical. Because the feedback byte from the neighboring nodes should be renewed according to the dynamic environment. In this case, the capture effect is supposed to effectively function.

Additionally, LiM models the redundancy optimization problem as a multi-armed bandit problem and maps a number of configurations to the corresponding actions in each sensor node. Furthermore, LiM employs a bandit learning scheme – Exp3 – in order to progressively optimize the efficiency of the network. This learning scheme investigates the selection of the "best" action for the dynamic environment, dramatically minimizing the redundancy of the communications while still maintaining a high reliability.

In general, there are two main phases in LiM: i) the greedy exploration phase and ii) the bandit learning phase. The former one is an exploration process where the "redundant" nodes can be discovered. This type of nodes acts as a concurrent transmitter in CT-based protocols. LiM attempts to seek these nodes and then keeps them staying in either receiving mode or sleeping mode in the network. Therefore, it is able to reduce redundancy and to save more energy. The latter phase is a reinforcement learning process. It has no conflict with data dissemination, which means, in this phase, LiM concurrently floods the information while progressively learns from the dynamics. In the following section, we explain these two phases of LiM in more detail.

3 DESIGN OF LIM

In this section, we detail the design aspects of LiM. We discuss the basis of LiM– concurrent transmissions and the reinforcement learning scheme, respectively.

3.1 Concurrent Transmissions

As derived from Glossy, LiM is based on concurrent transmissions, i.e., constructive interference and the capture effect. LiM adds an action scheme together with a feedback scheme to progressively learn the dynamics of the network. As a consequence, the content of a packet cannot be guaranteed to be identical all the time. The capture effect, however, is able to function well to correctly receive a packet with the stronger signal strength. In other cases, LiM similarly works with constructive interference as Glossy. Figure 1 shows the protocol stack of LiM. LiM operates IEEE 802.15.4 radios (i.e., $CC2420$) and is integrated with Glossy. As a result, LiM can be considered as an extension of Glossy, namely, it builds the learning scheme consisting of feedback and action selection on the higher layer of Glossy. The application layer can be further developed to meet the users' requirements. Later in this section, we explain the action and the feedback scheme in more detail.

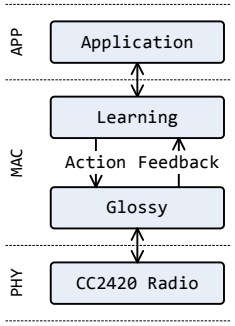


Figure 1: Protocol stack of LiM. LiM builds on Glossy and exploits an iterative learning scheme to select an action based on the feedback.

3.1.1 Number of Transmissions. By design, LiM maps four configurations of transmission times to four actions respectively: $Action\ 0$ stands for a node staying only in receiving (low-power listening) or sleeping mode, i.e., $N = 0$; $Action\ N$ ($N = 1, 2, 3$) means that a node works normally except setting the maximum transmission times to N, i.e., forwarding the received packet N times. In general, nodes in LiM exploit one of the above-mentioned four actions to effectively reduce the broadcast times in order to improve energy efficiency. Besides, the initiator in LiM is exempted from the action selection. Namely, the initiator maintains the default maximum number of transmission (i.e., $N = 5$, the same as Glossy), and does not execute neither the greedy exploration phase nor the bandit learning phase.

3.1.2 Frame Structure. To support the feedback scheme, LiM extends the frame structure in Glossy, by adding one byte for an exploring ID field and one byte for a feedback field, respectively. One example of a frame structure is illustrated in Figure 2. The frame length stands for the length of the whole frame. The header is a constant value, e.g., $0xA0$ in LiM. The length of the data (i.e., payload) in LiM can be adjusted according to the needs of different applications. By default, LiM sets the payload length to eight bytes. The exploring ID field is to disseminate the ID of the current exploring node – a node that receives the packet is able to notice whether it is the right time for itself to explore or to learn. While

a node experiences a packet loss, the feedback field is in use and updated to a negative-feedback value. If not, the feedback is not updated after the data packet has been received from the upper-level nodes, and hence, remains as a positive-feedback value. The relay counter is inherited from Glossy – for concurrent transmissions and time synchronization. A cyclic redundancy check (CRC) is an error-detection field to discover the accidental changes to the raw data while transmission in the air.

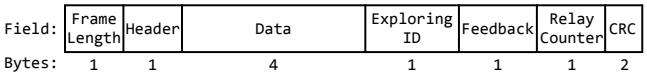


Figure 2: An example of an application-level frame structure in LiM. By design, the length of the data field (payload) is set to eight bytes in LiM. The exploring ID field is to notify the nodes in the network to proceed to different phases. The feedback field is to carry a response for the learning process.

3.2 Feedback Scheme

The feedback scheme is one key feature of LiM. Namely, only with the accurate feedback, sensor nodes can make the correct decision of choosing the appropriate action.

LiM assigns two types of feedback – a positive feedback ($0x01$) and a negative feedback ($0x00$). When in the exploration phase, if it is the time for a node to explore whether it is a "redundant" node or not, it stops forwarding the received packets in the current round. As a result, the nodes (e.g. in the lower level from the data source) that suffer a packet loss, update the feedback to a negative one. In the following round, while the exploring node recovers to forward, the nodes that lost a packet in the previous round rewrite the received packet with the renewed feedback byte and forward the packet to the neighboring nodes. Correspondingly, the exploring node i) receives the packet and ii) reads the feedback byte, and then iii) makes a decision on its own.

When in the bandit learning phase, the criterion is similar to the one in the exploration phase. If a node receives at least one packet, it does not update the feedback byte in the received packet. Otherwise, it renews the feedback byte to a negative value. The exploring node subsequently receives the packet and then updates the probability distribution of the action selection based on the received feedback. In a few cases, the capture effect could fail to work correctly due to the density of the nodes and the reason that the feedback byte is different, e.g., a positive feedback from "upside" and a negative one from "downside". In this case, LiM extends one extra listening time slot particularly for the packets from downside nodes to obtain a correct feedback. Figure 3 reveals the timeline of one single round in the flooding protocol.

3.3 Greedy Exploration

In addition to a bootstrap procedure, LiM comes to an exploration phase that can be considered as an initialization of the protocol. In this phase, LiM uses a greedy algorithm to explore the nodes that are not essential for transmitting (forwarding) the received packets in the network. Throughout this paper, we define this type of nodes as "*absorbing nodes*" that can always stay in either receiving mode or sleeping mode. Due to the special characteristics of

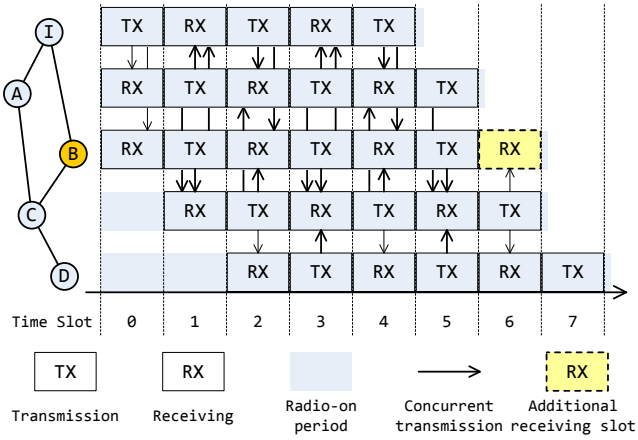


Figure 3: Time slots in one round of LiM with configuration of N = 3. Data packet is generated in and flowed from node I to all the others. LiM compels node B to extend one extra listening time slot for the exploring feedback, particularly for receiving the feedback from downside nodes.

these nodes, the number of absorbing nodes significantly affects the performance of the CT-based flooding protocols. On the one hand, reducing the number of absorbing nodes increases the concurrent transmitters, consequently strengthening the concurrent transmissions in the network. Based on the results in [4], however, the reliability (i.e., packet delivery ratio) degrades greatly with the number of synchronous transmitters. On the other hand, maintaining too many absorbing nodes might lead to a fragile network with a higher probability that nodes get disconnected while the environment dynamically changes.

LiM appropriately explores the absorbing nodes by considering the dynamical environment in the exploration phase using a greedy search algorithm. That is, the initiator firstly generates an exploring list containing all the node IDs in the whole network, and then disseminates each node ID in each packet in a number of rounds (e.g., 10 rounds for each single ID in LiM by default). As long as a node receives the packet containing the ID of its own, it acts as an absorbing node for the current round, i.e., it only receives the packets from others but does not forward any packet. Afterwards, the node works as a normal node in the next round, i.e., it continues to forward the received packets and meanwhile checks the feedback byte of the received data packet. Generally, as long as a node loses a packet in this phase, it updates the feedback byte in its received data packet and forwards it in the following round. In this case, each node attempts to explore whether it is an absorbing node Therefore, it is able to make a decision in the next round based on the feedback from its neighboring nodes.

Figure 4 demonstrates an example after greedy exploration in LiM. Previously, node B is in the first hop from the initiator, connecting node I and C (as shown in Figure 3). While in the exploration phase, node B finds out that it is not necessary for itself to forward the received packet to the next hop. Since with or without it, the neighboring nodes do not loss any packet. Hence, node B decides to act as an "absorbing node" – staying only in receiving mode or

sleeping mode. When a node decides to be an absorbing node, then it extends one more slot for listening on the occasion that it misses the packet in the first slot. This is because after the exploration phase, once being in the absorbing state, the node cannot forward a packet any more. Please note that, in the exploration phase, the initiator in LiM by default does not transmit any real data in the payload except the node ID, since the probability of packet loss is relatively high. However, if users can tolerate the loss, the initiator can be set to transmit real data also in this exploration phase. We show the reliability of LiM particularly in the exploration phase later in Section 4.

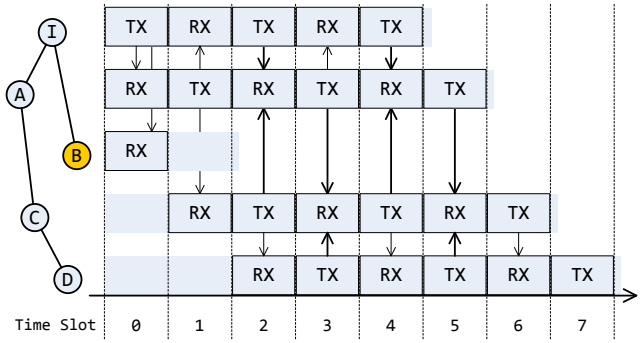


Figure 4: Nodes self-prune the connection links during the exploration phase. Node B then acts as an absorbing node and stays in receiving mode in a LiM round. The second time slot of node B attempts to relax the listening time, in case that it misses the packet in the first slot. In this example, node B saves the energy consumption of four slots compared to the other nodes.

3.4 Multi-armed Bandit Learning

In the following, we explain the details of the main learning algorithm in LiM. As different configurations are mapped to responding actions, we model the optimization problem as a multi-armed bandit problem. In order to overcome this problem, we use one algorithm from the set of multi-armed bandit learning algorithms – Exp3. In our case, the goal of the algorithm is to optimize the energy efficiency with reliability based on the policy of selecting transmission times for each sensor node.

Considering a process with K different actions, the Exp3 algorithm functions as shown in Algorithm 1, where γ is the exploration factor and w_i is the weight of each *action i*. $p_i(t)$ is the probability of selecting *action i* at round t, while T means the total number of iterations. At the beginning, the algorithm initializes the exploration parameter γ. This parameter adjusts the possibility that the algorithm attempts to explore other actions while a certain action has already achieved the highest probability. Next, the algorithm associates a weight with each action in order to give each action a probability to form a distribution over all actions.

After the exploration phase, the algorithm iterates T times the learning procedure in order to learn from the environment and to generate a better probability distribution to receive more accumulative rewards from the environment. In the learning procedure, the algorithm selects an *action i* based on the distribution $\mathcal{P}$ at first, and then receives a reward $x_{i_t}(t)$ from the environment. Thereafter, an estimated reward $\hat{x}_{i_t}(t)$ is calculated as $x_{i_t}(t)/p_{i_t}(t)$ to further include the influence of the probability on the reward. In the end, the weight of the sampled action is updated, while the weights of other actions ($w_j, \forall j \neq i_t, j \in \{1, \ldots, K\}$) remain unchanged. While the algorithm converges, the eventual probability distribution over different actions is considered to be the best strategy of maximizing the reward.

Algorithm 1 Exp3

1: **procedure** Initialization
2: initialize $\gamma \in [0, 1]$
3: initialize $w_i(1) = 1, \forall i \in \{1, \ldots, K\}$
4: for distribution $\mathcal{P}$,
5: set $p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}, \forall i \in \{1, \ldots, K\}$
6: **end procedure**
7: **procedure** Iteration T times
8: draw i_t according to $\mathcal{P}$
9: observe reward $x_{i_t}(t)$
10: define the estimated reward $\hat{x}_{i_t}(t)$ to be $x_{i_t}(t)/p_{i_t}(t)$
11: set $w_{i_t}(t+1) = w_{i_t}(t)e^{\gamma \hat{x}_{i_t}(t)/K}$
12: set $w_j(t+1) = w_j(t), \forall j \neq i_t$ *and* $j \in \{1, \ldots, K\}$
13: update $\mathcal{P}$:
14: $p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}, \forall i \in \{1, \ldots, K\}$
15: **end procedure**

To integrate Exp3 in LiM well, each action in this algorithm is associated with a possible configuration of a sensor node, which is mentioned in Section 3.1. In each iteration, the probability of selecting a certain action is calculated based on the feedback from previous broadcasting round. For instance, there are three actions ($K = 3$) in the learning procedure of LiM by design, i.e., *action* 1, 2, and 3. Respectively, *action* 1, 2, and 3 are mapped to three different configurations, where nodes transmit the received packet once, twice or three times. That is, if the randomly sampled *action i* is 1, then the node only transmits once in the current round. After the node receives the feedback, the weight of the corresponding action (i.e., *action* 1) is updated as shown in line 11 of the algorithm. The weights of other actions (i.e., *action* 2 and 3) stay the same.

In the final step, the distribution $\mathcal{P}$ is updated to prepare for the next iteration round according to the formula in line 14 of the algorithm. At this point, one learning iteration has been performed. This iteration phase continues until the number of iteration rounds reaches T. By design, LiM sets this value to $T = 200$, i.e., a fixed learning period for each node in LiM.

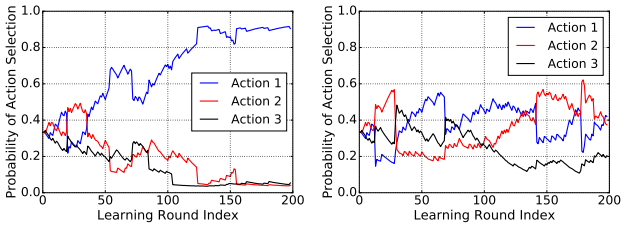
3.5 Implementation Aspects

In this part, we provide additional implementation aspects of LiM.

3.5.1 Destructive Action. While in the bandit learning phase, the nodes except the absorbing ones learn to make a decision based on the feedback they receive. With trying *action* 1, where a node only forwards the received packet once, receivers might miss the packet

so that the reliability of the whole network degrades. Because of the dynamics in the environment, this packet has higher possibility of getting lost since it is only transmitted once. Consequently, the nodes which are far away from the initiator would suffer a packet loss with relatively high probability. To avoid this negative effect, LiM learns conservatively to select *action* 1. Namely, if a node in LiM (i) gets a negative feedback of the exploring action in the previous round, and (ii) this specific action is *action* 1, then the node abandons selecting *action*1, i.e., it stops exploring *action* 1. The mechanism enables LiM making a relatively conservative decision of choosing *action* 1.

3.5.2 Not Enough Time for Convergence? Practically, the learning algorithm in a node may not always converge: The learning duration might not be long enough for the node to clearly distinguish the variance of the probability of selecting different actions. That is, in the end of the learning phase, the difference between each probability might not be large enough. Figure 5 demonstrates two cases of the convergence of the learning algorithm. In Figure 5(a), the probability of selecting *action* 1 converges and the node chooses *action* 1 as its final decision. Comparatively, as shown in Figure 5(b), the probability of choosing *action* 1 and *action* 2 is close. That means, the node might take a wrong decision according to the final probability. In case of this exception, to better maintain a high reliability instead of reducing the power consumption any more, LiM selects *action* 3 with the maximum transmission times $N = 3$. Note, that this is an example for a special case. In practice, LiM defines the learning round to be $T = 200$ in order to avoid the not-converged cases as often as possible.



(a) Converged case. (b) Not-converged case.

Figure 5: Two convergence cases of a learning phase in LiM. In (a), action 1 dominates in the end of the learning phase, while in (b), action 1 and action 2 still compete with each other in the end.

4 PERFORMANCE EVALUATION

In this section, we provide an evaluation of LiM based on a number of extensive experiments in a real-world testbed.

4.1 Methodology

In this work, we use the FlockLab testbed [9] for our experimental evaluation. FlockLab is located at ETH Zürich, Switzerland, and consists of 30 TelosB nodes inside and outside of an office building. For more details, we refer readers to Lim et al. [9] and the website[1] of FlockLab. Besides, the topology of the testbed is shown in Figure 6. We use 28 sensor nodes out of 30 existing ones (except node 201 and 202) in the testbed since they had not been constantly available during our experiments.

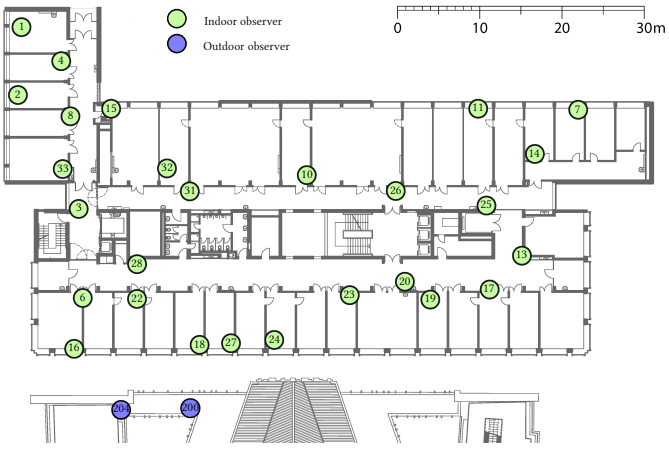


Figure 6: Deployment of sensor nodes in FlockLab.

To fairly evaluate the performance of the protocols, we use different nodes (i.e., node 1, 16 and 20) as initiator respectively in different scenarios. Besides, we vary the transmission power level as −7, −3, and 0 dBm in different scenarios. The default wake-up frequency of all protocols is set to 4 Hz, that is, a packet with 8 bytes payload is generated and transmitted by the initiator every 250 ms. Throughout this paper, we perform three independent runs for each experimental configuration. Each run lasts 45 minutes, in which there are over 10000 packets generated by the initiator and flooded to the whole network. All the results are averaged over these three runs, and the standard deviations are revealed by error bars.

4.1.1 Protocols. A number of the state-of-the-art protocols, integrate a specific application layer with Glossy, e.g., LaneFlood [15], It might not be fair enough to compare LiM to them since LiM is not application-specific such as LaneFlood. Therefore, in this the work, we only compare LiM to the our baseline Glossy in various scenarios. However, we believe that our protocol can be easily integrated with a specific application, for example, a data dissemination application. We are eager to compare LiM with the other relevant state-of-the-art protocols in the future.

4.1.2 Metrics. We focus on three key metrics to evaluate the performance of the related protocols, i.e., packet delivery ratio (PDR), radio-on time, and latency.

- *PDR*: The PDR is the ratio of the number of packets which are successfully delivered to a destination over the number of packets sent by the transmitter in an end-to-end communication. In most cases, PDR is the basic evaluation metric of a network, representing the reliability of the communication protocol.
- *Radio-on Time*: Radio-on time is the time duration that the radio is turned on in a single duty cycle, including the time for listening, receiving, and transmitting. Instead of considering the duty cycle – the portion of radio-on time over the total time, we directly take the radio-on

[1] www.flocklab.ethz.ch

time into account (note, that Glossy uses the same metric [2]), since the total time of each round is the same. It can be considered as an indicator of power consumption and describes the energy efficiency of the protocol. We measure radio-on time by using the software-based energy profiler [16] provided by Contiki.

- *Latency*: Latency is the time elapsed from the application on the initiator handing the packet to the MAC layer until the packet arrives at the other node's application. Therefore, latency in this paper represents the end-to-end latency on the application level. Minimizing end-to-end latency in random access networks is one of the key goals of protocol design, especially for mission-critical applications. In this paper, we measure latency based on the time-stamps of the serial outputs from all the sensor nodes.

4.2 Impact of the Number of Transmissions

In this part, we analyze how the performance metrics are affected by the number of transmissions (N) of a node during network flooding. We firstly run the experiments of our baseline Glossy and vary N as 1, 3, and 5.

As LiM flexibly tunes the number of transmissions (N) according to the learning experience.LiM starts with $N = 5$, as Glossy constantly sets N to 5 by default. In this part, for both protocols, we set node 1 – a node on the edge of the network – as the initiator, and use various transmission powers of −7, −3 and 0 dBm respectively, resulting in a WSN with different diameters.

Figure 7 revels the performance comparison between Glossy and LiM. For reliability, Glossy achieves extremely high PDRs even with various N values. LiM is able to maintain this advantage of Glossy – it achieves an average PDR of over 99.85% similar to the one of Glossy with $N = 1$. Moreover, LiM succeeds in reducing unnecessary broadcast redundancy, resulting in a notable decreased portion of radio-on time, compared to Glossy with $N = 3$ and 5. The radio-on time is even close to the one of Glossy with $N = 1$ by using −3 or 0 dBm as transmission power level. Even with a transmission power of −7 dBm, LiM is able to carry out a radio-on time similar to the one of Glossy with $N = 3$. For the flooding latency, LiM has to spend more time for data processing, decision making, probability calculation, and so on, consequently leading to an average latency of approximately 2 ms in most cases. Note that, in reality, Glossy with $N = 1$ may have a bootstrap problem and experience a highly fragile network according to our experience from the experiments that we carried out. We argue that LiM aims to progressively learn from the environment and thus makes a decision of N to progressively reduce the broadcast redundancy while maintaining acceptable levels of reliability and latency. In Section 4.4, we take a closer look at how LiM affects the number of transmissions.

4.3 Impact of the Topology

In this section, we evaluate LiM with different positions of the initiator. We change the position of the initiator (i.e., node 1, 16, and 20) to alter the flooding diameter of the network, making the topology different in each set of experiments. While the position of the initiator changes in the testbed, the logic network topology

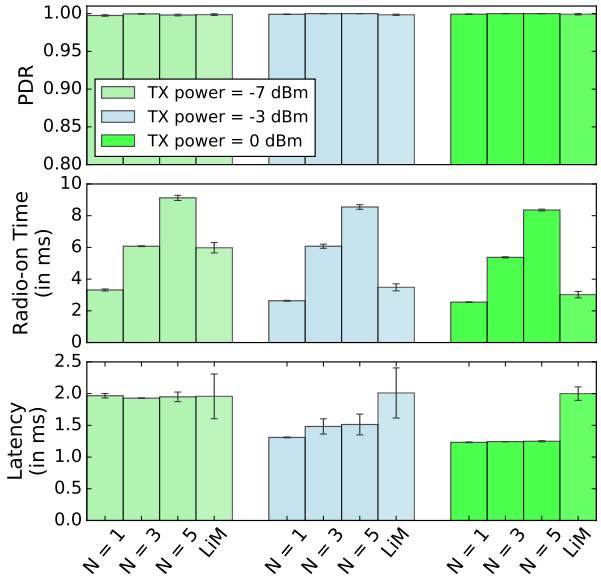


Figure 7: Performance metrics for Glossy with various N values and for LiM, respectively. Both protocols set node 1 as the initiator and use the transmission power of -7, -3 and 0 dBm. LiM inherits the advantages from Glossy in terms of high reliability with various levels of transmission power, while effectively reducing the radio-on time. The longer latency in LiM comes from the overall processing time in the bandit learning phase.

varies as well. Besides, we exploit the transmission power of −7 dBm to result in a WSN with approximately eight hops.

Figure 8 illustrates the results of LiM with various network topologies. For the reliability, LiM is able to achieve an average PDR of over 99.80% in all scenarios. The average radio-on time and latency change slightly along with the topology. However, LiM

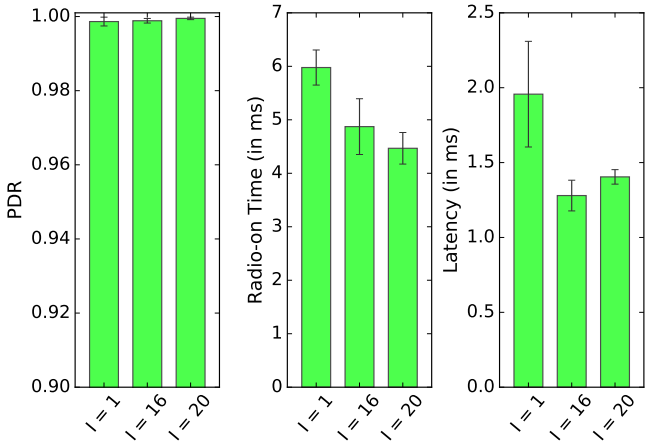


Figure 8: Performance metrics for LiM with various initiator positions using a transmission power of -7 dBm. Even with relatively weak link connections, the network can still provide high reliability, low radio-on time, and low latency.

maintains a less than 6 ms radio-on time and a less than 2 ms average end-to-end latency only with a transmission power of −7 dBm regardless of the topology.

4.4 Impact of the Exploration Phase

4.4.1 Absorbing Nodes. In general, the main part of redundancy reduction is contributed by the greedy exploration phase in LiM, where the absorbing nodes are discovered. These nodes stay in a receiving mode and do not forward the received packet after waking up from the sleeping mode. Figure 9 shows the average number of transmissions of each node in LiM from one experiment with node 1 as the initiator and 0 dBm transmission power. The nodes that have no transmission are actually the absorbing nodes, as the other nodes can be considered as a set of backbone nodes of the network in this case.

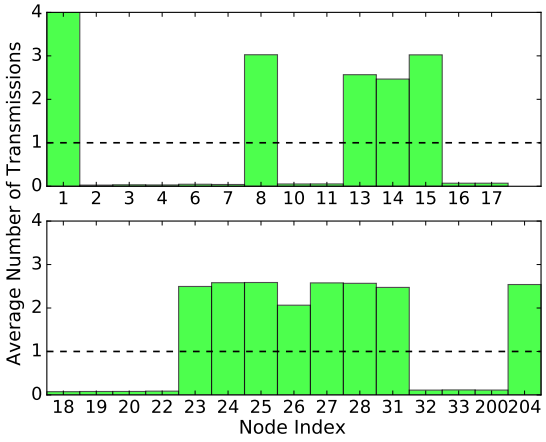


Figure 9: Average number of transmission with node 1 as initiator and 0 dBm transmission power. The dotted line shows the overall average number of transmissions. As LiM starts with N equal to 5, and progressively determines N, the average value of N in each node is not constant. After the learning phase, LiM eventually obtains an average N equal to 1.

4.4.2 Reliability Drop. Since nodes in the exploration phase exploit a temporary log-out strategy, the network reliability cannot be guaranteed to be 100%. However, LiM reserves 10 flooding rounds for each node to explore its role. Additionally, during these rounds, a node only logs out of the network one particular round (out of 10), where the node stays in receiving mode and does not transmit.

Consequently, assuming there are 30 nodes in a WSN and that nodes are well synchronized, then a node can lose 29 packets out of 300 in the worst-case, i.e., PDR equals 90.33%. Figure 10 illustrates the dynamically changing PDRs of all the nodes in the testbed along with the running time. As it shows, even though several nodes suffer a packet loss during the exploration phase, they are still able to maintain a high reliability afterwards.

4.5 Discussion

To summarize, in this section, we demonstrated the performance of two different protocols: LiM and our baseline protocol Glossy, respectively, in various evaluation scenarios. Our experimental results revealed that LiM effectively inherits the benefits from concurrent

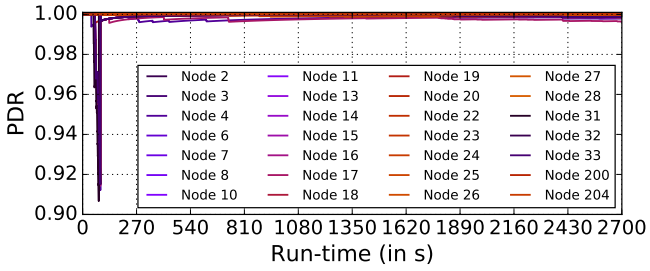


Figure 10: PDRs of all nodes in FlockLab changing over runtime. PDR drops during the greedy exploration phase while finding all the absorbing nodes, but it is still higher than 90%. In this case, users can decide whether to put important or dummy data in the payload during the exploration phase according to application-level requirements.

transmission. It delivers a high end-to-end reliability of over 99.50% with an average end-to-end latency of less than 2.5 ms in all cases. More importantly, LiM is able to decrease the radio-on time to less than 6 ms step-by-step, and significantly reduces broadcast redundancy. Even with different topologies, LiM is able to manage a high reliability with low end-to-end latency, while reducing unnecessary communication redundancy. Table 1 summarizes our experimental results.

5 RELATED WORK

5.1 Glossy-based Flooding Protocols

Network flooding is one of the most fundamental services in wireless sensor networks. It forms the basis for a wide range of applications and network operations. Glossy [2] provides a fast and efficient network flooding service by using concurrent transmissions in WSNs. By exploiting constructive interference and the capture effect on the physical layer, Glossy is able to get an average packet delivery ratio of 99.99% in real testbeds. Afterwards, Ferrari et al. add an application-level scheduler to construct a so-called Low-power Wireless Bus (LWB) [17]. LWB centrally schedules the data communication to support one-to-many, many-to-one, and many-to-many traffic patterns in WSNs. On the contrary, Chaos [4] builds on Glossy to achieve fast all-to-all data sharing in a distributed manner. Chaos further combines programmable in-network processing with concurrent transmissions in WSNs.

Meanwhile, Splash [18] builds a tree pipeline [19] on Glossy, thereby improving channel utilization. Furthermore, Pando [20] integrates fountain code with pipelining to overcome the long-tail problem of Splash. While Glossy disseminates one packet in each communication round, Splash and Pando are designed to deliver large data objects to all nodes in a network, e.g., for the purpose of reprogramming the WSN-based applications. Ripple [21] also relies on Splash and network coding techniques to further improve particularly in terms of network throughput.

Carlson et al. propose CXFS [6], a forwarder selection mechanism for concurrent transmissions. In CXFS, sensor nodes use a hop count in each packet to get their relative distance to each other. CXFS builds on Glossy and supports point-to-point transmissions while achieving high reliability, high energy efficiency, and high throughput. Moreover, Sparkle [22] selects subsets of

Table 1: Summary of experimental results. In all experimental scenarios, LiM maintains a best-effort end-to-end PDR and latency, taking advantage of concurrent transmissions. Besides, LiM decreases approximately 5 ms radio-on time compared to the default configuration of Glossy. The maximum value of radio-on time in Glossy is N = 5, while the minimum value is N = 1. LiM reduces the radio-on time by at least 3 ms compared to the default setting of Glossy (i.e., N = 5).

Protocol	Initiator ID	TX Power	PDR (%)	Radio-on Time (ms)	Latency (ms)
Glossy	Node 1	0 dBm	99.97 (max: 100.00 min: 99.88)	5.42 (max: 8.41 min: 2.53)	1.24 (max: 1.26 min: 1.23)
		-3 dBm	99.97 (max: 100.00 min: 99.88)	5.61 (max: 8.71 min: 2.61)	1.44 (max: 1.69 min: 1.30)
		-7 dBm	99.84 (max: 99.97 min: 99.66)	6.17 (max: 9.26 min: 3.25)	1.95 (max: 2.03 min: 1.89)
LiM	Node 1	0 dBm	99.91 (max: 99.98 min: 99.80)	3.02 (max: 3.22 min: 2.82)	2.00 (max: 2.12 min: 1.92)
		-3 dBm	99.84 (max: 99.92 min: 99.76)	3.48 (max: 3.69 min: 3.25)	2.01 (max: 2.34 min: 1.44)
		-7 dBm	99.86 (max: 99.95 min: 99.73)	6.00 (max: 6.33 min: 5.68)	1.95 (max: 2.33 min: 1.62)
	Node 16	0 dBm	99.97 (max: 99.99 min: 99.94)	3.89 (max: 4.70 min: 3.14)	0.91 (max: 1.06 min: 0.80)
		-3 dBm	99.84 (max: 99.91 min: 99.76)	3.64 (max: 4.07 min: 3.27)	1.53 (max: 1.58 min: 1.48)
		-7 dBm	99.89 (max: 99.96 min: 99.84)	4.87 (max: 5.38 min: 4.34)	1.28 (max: 1.38 min: 1.18)
	Node 20	0 dBm	99.91 (max: 100.00 min: 99.79)	2.96 (max: 3.11 min: 2.83)	0.94 (max: 1.12 min: 0.87)
		-3 dBm	99.93 (max: 99.94 min: 99.92)	3.11 (max: 3.20 min: 2.98)	1.18 (max: 1.23 min: 1.07)
		-7 dBm	99.95 (max: 99.98 min: 99.92)	4.47 (max: 4.77 min: 4.18)	1.40 (max: 1.44 min: 1.35)

nodes that participate in Glossy-based flooding. It also supports one-to-one communication. Similarly, LaneFlood [15] is built on Glossy and further integrates the forwarder selection scheme of CXFS with application-level network protocols in WSNs. LaneFlood thus supports one-to-one traffic, forwarder selection, and standard protocols in IoT such as TCP/UDP and constrained application protocol (CoAP). RTF [23] further extends Sparkle and exploits TDMA for data scheduling to improve reliability and energy-efficiency in point-to-point traffic. RFT identifies reliable relay nodes to limit the number of concurrently active neighbors to save more energy.

5.2 Bandit Learning in WSNs

Motamedi et al. in [24] propose a distributed multi-channel MAC protocol for wireless networks. It formulates the dynamic channel selection in wireless networks as a multi-armed bandit problem and derives optimal channel selection rules. The authors investigate the effectiveness of their protocol by using simulations. Thus, the effectiveness in real-world scenarios is therefore not clear. In [25], Kadono et al. propose a budget-limited multi-armed bandit algorithm, which is suitable for resource-constrained WSNs. It could limit sources to be retrieved when a relatively hard budget-limitation has been applied. By conducting simulations, they claim the proposed protocol outperforms the state-of-the-art.

A duty cycle learning algorithm (DCLA) is presented in [26] for IEEE 802.15.4 standardized WSNs. DCLA automatically adapts the duty cycle during run-time to minimize power consumption and to balance the packet delivery ratio and delay constraints of the application. It estimates the incoming traffic by collecting the network information during each active phase, and then uses a reinforcement learning framework to learn the best duty cycle at each beacon interval. Simulations demonstrate that the proposed scheme achieves the best overall performance for both, constant and event-based traffic, compared to existing IEEE 802.15.4 duty-cycled adaptation schemes.

In [27], the authors study long-term information collection in the WSN domain. Then, they propose a multi-armed bandit-based approach for the energy management problem in WSNs. They also describe a multi-armed bandit algorithm – Exp3 – that can be used to efficiently deal with the energy management problem. They show through simulations that their approaches improve the performance of the network by up to 120%.

5.3 Summary

Concurrent transmissions, a promising technique in this field, allow highly energy-efficient, low-power communication in WSNs. The technique has been developed and integrated with different standards and techniques. None of the state-of-the-art protocols, however, makes a great effort to apply an adaptive machine learning scheme to the concurrent transmissions. On the other hand, the bandit learning scheme has been exploited in the field of WSNs, for smart duty cycling, long-term energy management, and route selection. Most of the work are investigated by using simulations only. As a consequence, their effectiveness in real-world scenarios is not clear. LiM incorporates concurrent transmission with a bandit learning scheme in order to take advantages of both techniques. Meanwhile, LiM proves the feasibility of applying relatively "heavy" machine learning techniques to concurrent transmission for wireless networks in real-world applications. To the best of

our knowledge, LiM is the first primitive that integrates a machine learning scheme with concurrent transmissions, especially for low-power multi-hop WSNs. We believe that LiM is able to be further developed to robustly resist more adverse conditions in reality, e.g., with a channel hopping scheme, and to satisfy the requirements of the various applications.

6 CONCLUSION

This paper introduces LiM, a machine learning-based flooding protocol for low-power duty-cycled WSNs. LiM applies a bandit learning scheme in CT-based flooding, thereby, benefiting from both. Concurrent transmissions ensure LiM a highly reliable communication with low end-to-end latency and low energy cost. Machine learning brings the adaptation ability to deal with the dynamics of the environment, thereby further improving energy efficiency. We implement our protocol in Contiki and evaluate it with extensive experiments in FlockLab.

Our experimental evaluation shows that LiM achieves less radio-on time, and – as a consequence – it greatly improves energy efficiency of the network. Meanwhile, LiM manages a more than 99.50% average end-to-end reliability and less than 2.5 ms average end-to-end latency in all experiments in the testbed. Furthermore, with its learning ability, LiM maintains a flexible adaptation to the dynamics of the network, while compared to the baseline protocol Glossy. To sum up, LiM inherits the benefits from concurrent transmissions and a machine learning scheme, outperforming our baseline protocol Glossy in the light of energy efficiency while maintaining a high end-to-end reliability and low latency. In the future, we plan to extend LiM to the frequency domain, e.g., adding a channel hopping strategy, to enhance the robustness of the protocol.

7 ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. We also thank the Computer Engineering Group at ETH Zürich for providing the FlockLab testbed. This research was supported by the German Research Foundation through the Research Training Group DFG-GRK 1765: System Correctness under Adverse Conditions (SCARE, www.uni-oldenburg.de/scare).

REFERENCES

[1] J. Lu and K. Whitehouse. Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks. In *Proceedings of the 28th Conference on Computer Communications (INFOCOM)*, pages 2491–2499, April 2009.

[2] 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*, pages 73–84, April 2011.

[3] Y. Tseng, S. Ni, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.*, 8(2/3):153–167, March 2002.

[4] 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 '13, pages 1:1–1:14, New York, NY, USA, 2013. ACM.

[5] V. S. Rao, M. Koppal, R. V. Prasad, T. V. Prabhakar, C. Sarkar, and I. Niemegeers. Murphy loves ci: Unfolding and improving constructive interference in WSNs. In *Proceedings of The 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2016)*, pages 1–9, April 2016.

[6] D. Carlson, M. Chang, A. Terzis, Y. Chen, and O. Gnawali. Forwarder selection in multi-transmitter networks. In *Proceedings of the 2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 1–10, May 2013.

[7] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[8] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th IEEE Annual International Conference on Local Computer Networks*, LCN '04, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[9] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, pages 153–166, New York, NY, USA, 2013.

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

[11] K. Leentvaar and J. H. Flint. The capture effect in FM receivers. *IEEE Transactions on Communications*, 24(5):531–539, May 1976.

[12] T. Mitchell. *Machine Learning*. McGraw Hill, 1997. ISBN: 0070428077.

[13] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th IEEE Annual Foundations of Computer Science*, pages 322–331, Oct 1995.

[14] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[15] M. Brachmann, O. Landsiedel, and S. Santini. Concurrent transmissions for communication protocols in the internet of things. In *Proceedings of the 41st IEEE Conference on Local Computer Networks (LCN)*, pages 406–414, Nov 2016.

[16] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, pages 28–32, New York, NY, USA, 2007. ACM.

[17] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pages 1–14, New York, NY, USA, 2012. ACM.

[18] 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'13, pages 269–282, Berkeley, CA, USA, 2013. USENIX Association.

[19] 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 '10, pages 15–28, New York, NY, USA, 2010. ACM.

[20] 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 '15, pages 365–378, New York, NY, USA, 2015. ACM.

[21] 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)*, pages 1–9, June 2015.

[22] D. Yuan, M. Riecker, and M. Hollick. Making 'Glossy' networks sparkle: Exploiting concurrent transmissions for energy efficient, reliable, ultra-low latency communication in wireless control networks. In *Proceedings of the 11th European Conference on Wireless Sensor Networks - Volume 8354*, EWSN 2014, pages 133–149, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[23] J. Zhang, A. Reinhardt, W. Hu, and S. S. Kanhere. RFT: Identifying suitable neighbors for concurrent transmissions in point-to-point communications. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '15, pages 73–82, New York, NY, USA, 2015. ACM.

[24] A. Motamedi and A. Bahai. MAC protocol design for spectrum-agile wireless networks: Stochastic control approach. In *Proceedings of the 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, pages 448–451, April 2007.

[25] Y. Kadono and N. Fukuta. LAKUBE: An improved multi-armed bandit algorithm for strongly budget-constrained conditions on collecting large-scale sensor network data. In *Proceedings of the 13th Pacific Rim International Conference on Artificial Intelligence*, pages 1089–1095. Springer International Publishing, 2014.

[26] R. D. P. Alberola and D. Pesch. Duty cycle learning algorithm (DCLA) for IEEE 802.15.4 beacon-enabled wireless sensor networks. *Ad Hoc Networks*, 10(4):664 – 679, 2012. Advances in Ad Hoc Networks (II).

[27] L. Tran-Thanh, A. Rogers, and N. R. Jennings. Long-term information collection with energy harvesting wireless sensors: a multi-armed bandit based approach. *Autonomous Agents and Multi-Agent Systems*, 25(2):352–394, 2012.