Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 7651861, 19 pages
https://doi.org/10.1155/2019/7651861

Research Article

Oppo-Flood: An Energy-Efficient Data Dissemination Protocol for Asynchronous Duty-Cycled Wireless Sensor Networks

Hyeonsang Cho and Jungmin So

Department of Computer Science and Engineering, Sogang University, Seoul 04107, Republic of Korea

Correspondence should be addressed to Jungmin So; jso1@sogang.ac.kr

Received 23 July 2019; Accepted 15 November 2019; Published 2 December 2019

Academic Editor: Rüdiger C. Pryss

In this paper, we propose a data dissemination protocol for asynchronous duty-cycling wireless sensor networks. In an asynchronous duty-cycling network, each node independently selects its wake-up time. In this environment, data dissemination becomes energy consuming, because broadcasting a packet does reach all neighbors but only the neighbors that are awake at the time. A node can forward its packet to all neighbors by continuously transmitting the packet for a whole wake-up interval, but it leads to high energy consumption and high dissemination delay. The idea proposed in this paper is to use opportunistic forwarding, where each node forwards the packet to a neighbor that wakes up early and receives the packet. Each node forwards the packet, as long as there is a neighboring node that has not received the packet yet. The main benefit of this opportunistic forwarding-based dissemination is that every time a packet is disseminated, it may take a different path to reach the nodes. At the beginning of dissemination, a sender needs to transmit for a very short duration of time because there are plenty of neighboring nodes to receive the packet. As more nodes receive the packet, the transmit duration of the sender becomes longer, thus consuming more energy. Since the order of dissemination is different every time, energy consumption is naturally balanced among the nodes, without explicit measures. Through extensive simulations, we show that the proposed protocol achieves longer network lifetime and shorter dissemination delay compared to other dissemination protocols in various network environments.

1. Introduction

A wireless sensor network is a type of wireless network that is often used for environmental and industrial monitoring. A typical wireless sensor network consists of small battery-powered devices deployed throughout an area. Each device has one or multiple sensors for collecting data and a wireless transceiver for sending and receiving messages. Most wireless sensor networks are in the form of a multihop ad hoc network where packets originated from one node may travel through multiple links via other nodes in order to reach its destination, and there is no infrastructure. Wireless sensor networks are often expected to operate for a long time without human intervention, so conserving energy consumption is one of the most important issues in designing a network. For that reason, many wireless sensor networks use duty cycling; a technique where nodes in the network switch between sleep state and active state. A node in a sleep state cannot send or receive packets but consumes much less energy compared to when a node is in an active state. A node may operate on a very low duty cycle such as 1%, which means the node is in active state for 1% of the time and stays in sleep state for 99% of the time.

Since nodes cannot transmit or receive packets in sleep state, forwarding packets becomes tricky; when a node wants to transmit a packet, its intended receiver may be in sleep state and cannot receive the packet. A medium access protocol should address this issue, and there are two major approaches: synchronized and asynchronous protocols. In a synchronous MAC protocol, a sender should be aware of the receiver's wake-up schedule and transmit when the receiver is awake and ready to receive the packet. Some synchronized MAC protocols make all nodes wake up at the same time, whereas other protocols allow different schedules but nodes exchange messages with neighbors to find out their schedules. Synchronized protocols have several issues. First, time synchronization is necessary, which may often require extra message overhead. Second, when a node wakes-up, multiple nodes may transmit their packets to the node simultaneously, resulting in collision and packet loss. A collision avoidance technique such as random backoff should be applied in order to avoid collision. Third, if nodes are allowed to operate on different schedules, further message overhead is required to exchange information between neighbors.

On the other hand, in an asynchronous MAC protocol, nodes are not time-synchronized, and they do not know when the neighbors will wake up. In a well-known asynchronous MAC protocol called BoX-MAC, a node wanting to send a packet wakes up and transmits the packet repeatedly. When the receiver wakes up according to its schedule, the node receives the packet and sends an acknowledgment (ACK) to the sender. On receiving the ACK, the sender finally stops transmitting and goes back to the sleep state. Asynchronous MAC protocols are simpler to implement than synchronized protocols, because asynchronous protocols do not require control overhead to maintain synchronization. In this paper, we consider wireless sensor networks that operate on asynchronous MAC protocols.

Data collection is a major application of wireless sensor networks, where information flows from sensor nodes to one or multiple sink nodes. A data collection tree can be established from the sink node to all the sensor nodes, and packets are forwarded from the nodes to the root node. For data collection, a typical form of transmission is unicast, but sometimes multicast or anycast are also used. Although data collection is a typical application of a wireless sensor networks, data dissemination is also an important operation. In contrast to data collection, information flows from one node (a sink or a sensor node) to all sensor nodes in the network. Data dissemination is used to update parameters and configurations, download code to the sensor nodes, or issue network-wide commands if the nodes are equipped with actuators.

Most dissemination protocols take one of the two approaches: Flooding or Tree-based. In Flooding, a node forwards a packet to all of its neighbors, and each node receiving the packet also forwards it to its neighbors. In a Tree-based protocol, a dissemination tree is established in the network, rooted at the source node. Nodes forward the packet to their child nodes, and thus the packet is forwarded along the tree structure. In an asynchronous duty-cycling sensor network, both approaches have drawbacks, as we discuss further in Section 3. A Flooding-based approach is reliable, but creates too many redundant packets that lead to unnecessary energy consumption. A Tree-based approach shows the energy-hole problem, in which energy drains faster in nodes near the source than nodes far away from the source. In this paper, we propose a new data dissemination protocol suited for an asynchronous duty-cycling network, which uses opportunistic forwarding for dissemination. Opportunistic forwarding has been shown to work well with asynchronous duty-cycling for data collection applications [1, 2]. Here, we use opportunistic forwarding for data dissemination, with the goal of achieving similar benefits such as low delay and low energy consumption. In the proposed protocol, each node forwards the packet to one of its neighbors, who wakes up early and receives the packet first. A node uses two-hop neighbor information in order to find out if the node is an articulation point and forwards the packet to multiple nodes if necessary. The benefit of the proposed protocol is that depending on when the dissemination occurs, the packet is forwarded along a different path because of the opportunistic forwarding. This balances load and energy consumption among nodes without any explicit measures. The simulation results show that the proposed protocol can achieve longer network lifetime compared to Flooding and Tree-based protocols in various network environments.

As discussed in Section 2, there have been a few protocols that use opportunistic forwarding in data dissemination [3, 4]. However, they require wake-up schedules of neighbors and link quality information, which is cumbersome to maintain and requires message overhead. To use neighbor wake-up schedules, notification should be made whenever a node changes its wake-up schedule. It is even harder to maintain and exchange link quality information, because many packets need to be actually transmitted on the link in order to estimate the average link quality. Our proposed protocol does not require these information and thus can work on top of BoX-MAC. We believe it is important for practicality of the protocol because BoX-MAC is a widely used protocol.

The rest of the paper is organized as follows. In Section 2, we discuss the existing work that considers data dissemination in multihop wireless networks. In Section 3, we briefly introduce relevant system and protocols and motivate our design of a new data dissemination protocol. In Section 4, we describe the proposed protocol, Oppo-Flood, in detail. In Section 5, we evaluate the performance of Oppo-Flood, comparing with other existing protocols. Finally, in Section 6, we conclude the paper with remarks for future work.

2. Related Work

Data dissemination is an important operation in wireless sensor networks as well as other multihop networks such as wireless ad hoc networks. Naturally, many dissemination protocols have been proposed for different network environments and application requirements. Earlier protocols were developed for non-duty-cycling networks where nodes are mostly in active mode and ready to receive packets. Trickle [5] and Deluge [6] are epidemic type dissemination protocols where each node occasionally broadcasts its data to its neighboring nodes. If a node finds out that a neighboring node has a newer version of data, the node sends a request in order to acquire the new data. Redundant transmissions are suppressed by nodes overhearing each other, and broadcast rate is dynamically adjusted depending on node density. Kyasanur et al. [7] proposed Smart Gossip, a probabilistic forwarding scheme to reduce redundant transmissions caused by Flooding. The probability of forwarding is determined based on the importance of the nodes. Stann et al. [8] proposed RBP, which controls number of retransmissions depending on density of nodes in the vicinity in order to balance the reliability and energy consumption. Huang and Setia proposed CORD [9], which minimizes number of transmissions by establishing connected dominating set on the network. Nodes that are not involved in transmitting or receiving data are put to sleep in order to save energy. Sprinkler [10] adopts a similar strategy, which divides the area into a virtual grid of equal sized planes and builds a connected dominating set using the grid. Zhu et al. [11] proposed Collective Flooding, which uses collective ACKs to reduce the number of ACKs. Using correlation between the links, the sender estimates whether a packet is successfully received by a node, from an ACK sent by another node. Dong et al. [12] proposed ECD, where packet senders are selected according to link quality and packet sizes are dynamically configured in order to improve transmission efficiency. These protocols are not suitable for asynchronous duty-cycling networks, because they assume the neighboring nodes will always receive the packet if the link quality between the nodes allows successful reception.

Data dissemination protocols have also been proposed for networks with synchronized duty-cycling networks. Miller et al. [13] proposed a broadcast protocol where all nodes follow the same active-sleep cycle, but some nodes wake up in the sleep cycle to send packets and some nodes decide to remain active in the sleep period. These two events occur with probability p and q which establishes the tradeoff between reliability, packet delay, and energy consumption. Lu and Whitehouse proposed Flash Flooding [14], which exploits capture effects to allow concurrent transmissions and thus rapid Flooding. Ferrari et al. proposed Glossy [15] which exploits constructive interference by allowing time-synchronized concurrent transmissions, leading to successful packet receptions without the capture effect. Splash [16] builds on top of Glossy and adds tree pipelining technique to achieve fast and energy-efficient Flooding. A reverse path of data collection tree is used as dissemination paths, and odd-hop nodes and even-hop nodes take turns to transmit and receive Flooding packets. Pando [17] improves the performance of tree pipelining by applying Fountain coding in order to defend against unreliability of constructive interference. Since every encoded packet contains new information of the original data, receivers become insensitive to loss of individual packets.

The environment we consider in this paper is an asynchronous duty-cycling network, where each node independently switches between active and sleep states. Many dissemination protocols were proposed for this type of network as well. Sun et al. [18] proposed ADB, a protocol designed to work on top of RI-MAC [4]. In ADB, the broadcasting is done through a tree, but a node can opportunistically delegate its transmission to one of its child nodes. For example, suppose node A has node B and C as its child nodes. If node A finds out that the link condition between B and C is better than the link condition between A and C, A can delegate its transmission to B so that B can forward its packet to C on behalf of A. Lai et al. proposed Hybrid-cast [19], which uses quorum-based wake-up scheduling and delivery deferring to reduce the number of transmissions. In Hybrid-cast, each node broadcasts a beacon at the beginning of a wake-up slot. On receiving a beacon, the data sender defers its transmission for a period of time hoping that more neighbors wake up during that time so that one broadcast transmission could be received by multiple neighboring nodes. Hong et al. [20] proposed an algorithm for building a connected dominating set on the network to minimize the number of broadcast transmissions. The proposed algorithm considers the fact that a node may need to transmit multiple times to cover its neighbors, because they may wake up at different times. Tang et al. [21] and Duc et al. [22] have followed the same approach but have improved the performance of CDS-based broadcasting.

Guo et al. [23] proposed a protocol where a flooding tree is constructed on the network considering link correlation. The tree is established so that nodes choose a common parent node if the links between the nodes and the parent node are highly correlated. After selecting a common parent, the nodes wake up at the same time to receive the broadcast packet from the parent node. Only one node sends back an ACK in order to mitigate ACK implosion and energy consumption. Zhao et al. [24] have followed a similar approach but their protocol is designed to achieve 100% reliability required for WSN reprogramming. Wang and Liu [25] proposed a protocol where the broadcast problem is formulated as a shortest path problem in a time-coverage graph. A centralized and distributed version of the protocol is presented which calculates the shortest path considering node wake-up schedules. Xu et al. [26] proposed a protocol where nodes extend their active time in order to overhear broadcast packets from neighbors, thereby reducing packet delay and energy consumption. Niu et al. [27] proposed a protocol for building a flooding tree in the network. The protocol initially builds an ETX- (Expected Transmission Count-) based shortest path tree. While operating, nodes can dynamically reselect the best parent node based on link quality, while ensuring loop-free property of the flooding tree. Guo et al. [3] proposed a protocol which is based on a tree structure, but allows opportunistic transmissions to paths outside the tree. Due to unreliable links, forwarding packets along the tree may cause long packet delay. In order to reduce the delay, a node forwards to a neighboring node outside the tree if it is statistically probable that the receiving node will receive the packet earlier than receiving the packet from its parent node. Han et al. [28] proposed a protocol where nodes can adjust their transmission power. Since the network topology changes according to transmission power, the goal is to find a valid dissemination tree that minimizes the total transmission power consumed. Sutton et al. proposed Zippy [29], a protocol for infrequent on-demand flooding. The protocol uses an additional low-complexity transmitter and a receiver that are always on so that nodes can quickly wake-up when the flooding needs to take place. Xu et al. [30] proposed a dissemination protocol with the focus of balancing the transmission load in the network. The flooding tree is built and adjusted so that each node has similar number of child nodes and thus similar transmission load. Cao et al. proposed Chase [31], a broadcast service that allows concurrent broadcast transmissions using random interpreamble packet interval adjustments. Chase classifies signal patterns to detect whether concurrent transmission is possible or not and extends radio-on time if successful packet delivery is expected.

Although many dissemination protocols have been proposed for asynchronous duty-cycling WSNs, most of them require and utilize knowledge of wake-up schedules in 1-hop or 2-hop neighborhood. When a node needs to send a packet, it wakes up just before the scheduled wake-up time of the receiver and transmits the packet. Acquiring wake-up schedule needs time synchronization among nodes, as well as control overhead to exchange schedules occasionally. ADB [18] does not require nodes to know wake-up schedules of neighboring nodes. However, quality of links between every pair of nodes must be measured, in order to decide whether a node should delegate its transmission or not. Acquiring accurate link quality is challenging, especially when nodes are on a low-duty cycle [2]. Also, ADB does not consider load balancing and may suffer from the energy hole problem [32]. Different from existing works, our target environment is an asynchronous duty-cycling network where nodes do not know the wake-up schedule of neighboring nodes. Under this environment, our goal is to design a dissemination protocol that is energy-efficient and load-balancing, in order to achieve long network lifetime.

3. Preliminaries

3.1. BoX-MAC. BoX-MAC [33] is the default MAC protocol implemented in TinyOS-2.x, an operating system for sensor devices. BoX-MAC operates on top of a wireless sensor network system where nodes duty-cycle asynchronously. Whenever a node wants to transmit a packet to a specific destination (unicast), the node wakes up and performs carrier sensing on the channel. If the channel is idle, the node transmits a packet repeatedly, with short intervals in the middle of each transmission (as shown in Figure 1). Other nodes wake up according to their schedules. Whenever a node wakes up, it first checks the channel to see if energy is detected on the channel. If not, the channel is idle, so the node immediately goes back to sleep (*idle receive check*). If there is energy, the node tries to receive a packet. When a node starts receiving the packet, it checks the packet header to determine whether it should continue receiving the packet or not. If the packet is not intended for the node itself, the node aborts reception and goes back to sleep (*invalid packet reception*). Finally, if the node is the intended receiver, it continues receiving the packet until the end and sends back an ACK to the sender (*valid packet reception*). When the sender receives an ACK, it stops transmission and goes back to sleep. If we assume that every node runs on the same wake-up interval of w seconds, the sender has to wait for w seconds in the active mode before its destination node wakes up, in the worst case. This sender "wait time" is the most significant source of energy consumption in asynchronous duty-cycling networks.

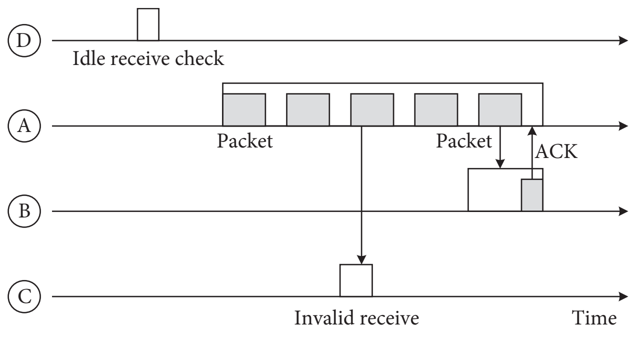


Figure 1: Behavior of BoX-MAC. Node A has a packet to send to B so it wakes up and sends the packet repeatedly. When node B wakes up according to its schedule, it receives the packet and sends an ACK to node A. On receiving the ACK, node A stops transmission and goes back to sleep. Node D wakes up before node A starts transmitting, so it goes back to sleep after checking the channel. Node C wakes up and starts receiving the packet, but later finds the packet to be invalid (not to be received) and goes back to sleep.

3.2. Data Dissemination in Asynchronous Duty-Cycling WSNs. Here, we look at the basic approaches for disseminating data in asynchronous duty-cycling WSNs. As mentioned earlier, our assumption is that the nodes do not know the wake-up schedules of neighbors, which is the same assumption for the BoX-MAC. Also, this assumption makes our protocol design different from most approches taken in existing protocols, as discussed in the previous section. We look at the two basic approaches to data dissemination: network wide Flooding and Tree-based dissemination.

3.2.1. Flooding-Based Data Dissemination. The basic flooding technique does not require any structures in the network. Initially, the source node broadcasts a packet to all of its neighbors. On receiving a packet, a node also broadcasts the packet to all its neighbors. In order to avoid sending duplicate packets, the broadcast packet has a sequence number in the header. If a node receives a packet that has been received and forwarded previously, the node simply discards the packet. In an asynchronous duty-cycling network, nodes wake-up for a short duration of time periodically and their wake-up times are different. In order to reach all neighbors, a node has to transmit the packet repeatedly for a whole wake-up interval. If nodes use different wake-up intervals, a node should transmit the packet for the longest wake-up interval used by its neighboring nodes. Since receiving nodes do not reply with ACKs for broadcast packets, the sender just decides to transmit for a certain duration of time and finishes transmission without knowing who received the packet. If link conditions are bad or packet collisions occur, a single wake-up interval may not be sufficient, and the sender may need to transmit longer in order to have all neighbors receive the packet.

It is well known that when a packet is flooded, too many duplicates are created if all nodes forward the packet to all neighbors [34]. The level of redundancy depends on the network density. We have conducted a preliminary experiment, where we have deployed a number of sensor devices in an area and had one of the node periodically send packets to be disseminated to all the nodes in the network (the simulation setup is the same as what we used in Section 5, and it will be described there in detail). We have varied number of nodes to study the impact of node density in the data dissemination performance. Each packet has a sequence number, and a node only broadcasts once for a packet with a particular sequence number. Whenever a node starts transmitting, it continues for a whole duration of wake-up interval, which is set to 1 second in this experiment. The result is shown in Figure 2. Figure 2(a) shows that regardless of node density, dissemination ratio is almost 100%, since each node broadcasts packets for a whole wake-up interval. However, as shown in Figure 2(b), average energy consumption of the nodes increase with node density, and that is due to the increased number of duplicates shown in Figure 2(c); nodes spend more time in active mode receiving unnecessary packets.

A natural intuition that follows from this result is to reduce the duration of broadcast; a node does not transmit its packet for a whole duration of time. Only a subset of neighbors will receive the packet, but if there is enough number of neighbors, duplicate copies of packets forwarded by them could reach the whole network. To see if this is true, the next experiment is conducted. For a fixed node density, we have varied the transmission duration. The result is shown in Figure 3. Similar to the previous experiment, the wake-up interval of all nodes is 1 second.

Figure 3(a) shows that the dissemination ratio decreases when the transmission duration becomes short. If we need to achieve a certain level of dissemination ratio, the proper transmission duration depends on node density; when the network is dense, each node can transmit shorter and still achieve high dissemination ratio. Figures 3(b) and 3(c) show that energy consumption per node is higher for dense networks due to duplicate packets. Thus, it is important to control transmission duration based on network density in order to improve energy efficiency and achieve long network lifetime.

In the previous experiments, we assumed that all links are reliable. The received signal strength is calculated based on a path loss model, and a packet is always received correctly if the signal-to-noise ratio (SNR) is higher than a certain threshold. In reality, the nodes could be deployed in an environment where packet loss frequently occurs due to various environmental reasons such as fading. In order to model that, we have assigned loss rates to links. If a link between a pair of nodes has a loss rate of 30%, it means that 30% of the packets that are transmitted between the pair are dropped. In the next experiment, we have varied the link loss rate to see its impact on dissemination ratio and energy consumption (all links have the same loss rate). The result is shown in Figure 4.

We can observe from Figures 4(a) and 4(b) that when link loss rate increases, the dissemination ratio drops while the energy consumption stays similar. When transmission duration is 1 second (which is the same as the wake-up interval), the dissemination ratio is near 100% until the link loss rate goes over 30%. Then, the dissemination starts to drop. This means that, with lossy links, a node may need to transmit its packet longer than a wake-up interval, in order to achieve a certain level of reliability. When the transmission duration is 0.3 seconds, the dissemination ratio is significantly degraded as the link loss rate goes up. Suppose the application has a reliability requirement, such as 90% dissemination ratio. If link loss rate is 30%, we can achieve the reliability level with transmission duration of 0.6 seconds. But if link loss rate is 50%, a longer transmission duration is required. This experiment shows that a proper transmission duration depends on node density and link loss rate. The difficulty of using this approach is that since ACKs are not used, the network does not know how many nodes have successfully received the broadcast packet. Without feedback from the receivers, a node cannot adjust its transmission duration in order to improve reliability or reduce energy consumption.

3.2.2. Tree-Based Data Dissemination.

A Tree-based dissemination protocol builds a dissemination tree in the network, and the packets are forwarded along the tree. The dissemination tree could be a reverse of a collection tree if the collection sink and the dissemination source is the same node. In a typical Tree-based dissemination protocol, a parent node unicasts the packet to each of its child nodes. Since unicast packets are replied by ACKs, a parent node knows which of the child nodes have successfully received the packet. In order to achieve 100% reliability, each node simply needs to make sure that all of its child nodes receive the packet. Then, all nodes will receive the packet, except the ones that are disconnected from the network.

Provided that the network topology does not change frequently and the link conditions are not bad, a Tree-based approach is expected to consume less energy compared to a flooding-based approach, because nodes do not need to transmit the packet for a whole wake-up interval and leaf nodes do not need to transmit packets at all. However, provided that the nodes are uniformly distributed, a Tree-based dissemination could lead to the energy hole problem [32], where energy drains faster in nodes that are close to the source than nodes far away from the source. It is because nodes near the source tend to have more child nodes, and thus, they need to transmit packets for longer duration of time.

We have conducted a preliminary experiment on the Tree-based dissemination approach. Varying the number of nodes, we have measured dissemination ratio and average and maximum node energy consumption. The maximum node energy consumption is related to the network lifetime, when we define network lifetime as time until all of the energy drains in the first node [35]. We have experimented three variations of Tree-based dissemination. First, “Tree” is the basic tree-based protocol in which a broadcast packet is transmitted from the source node, and each node receives the broadcast message to select a parent node based on the hop distance from the source node. The tree-building phase is not included in the energy consumption measurements. Second, “Tree-Balanced” is a protocol where if a node has

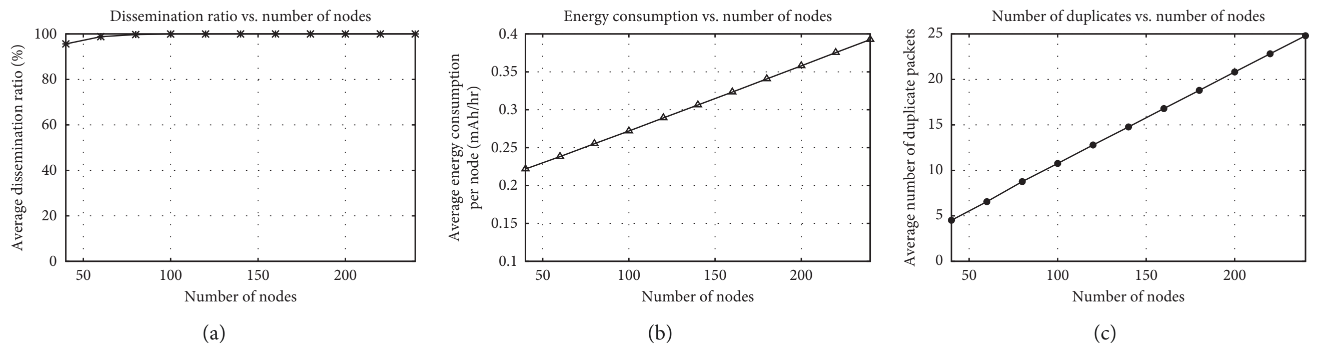


FIGURE 2: Preliminary experiment results with flooding-based dissemination. In this experiment, number of nodes is varied from 40 to 240. Tx duration is fixed at 1 second. Links are assumed to be reliable. (a) Dissemination ratio. (b) Energy consumption. (c) Number of duplicates.

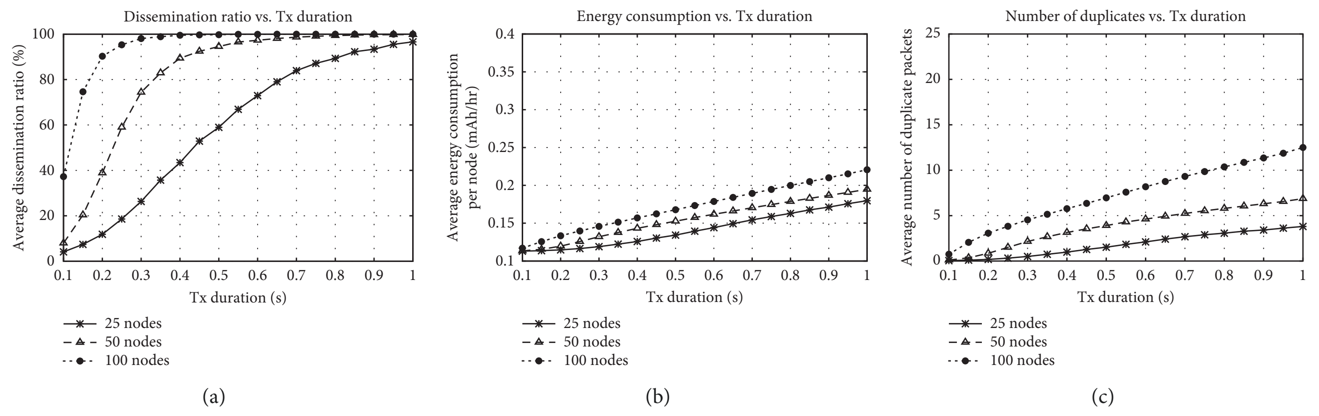


FIGURE 3: Preliminary experiment results with flooding-based dissemination. In this experiment, Tx duration is varied from 0.1 to 1 second. For number of nodes, 25, 50, and 100 nodes are used. Links are assumed to be reliable. (a) Dissemination ratio. (b) Energy consumption. (c) Number of duplicates.

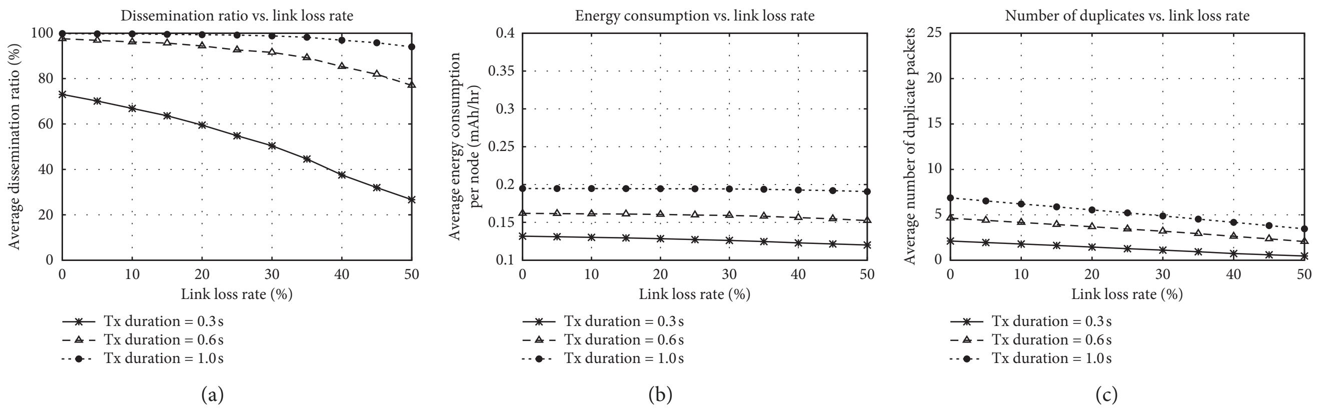


FIGURE 4: Preliminary experiment results with flooding-based dissemination. In this experiment, link loss rate was varied from 0 to 50%. Number of nodes was fixed at 100. For Tx duration, 0.3, 0.6, and 1 seconds were used. (a) Dissemination ratio. (b) Energy consumption. (c) Number of duplicates.

multiple candidate parents with the same hop distance from the source node, it selects a parent node which has the minimum number of child nodes. This is to balance the number of child nodes, which will lead to balanced load among nodes. Finally, "Tree-Dynamic" is a protocol where each node periodically reselects its parent node, which has the maximum residual energy among candidate parent nodes. This method is expected to balance energy consumption among nodes. The result is shown in Figure 5. First, all the tree-based protocols achieve 100% dissemination ratio. When looking at the average and maximum energy consumption, we can observe that there is a gap

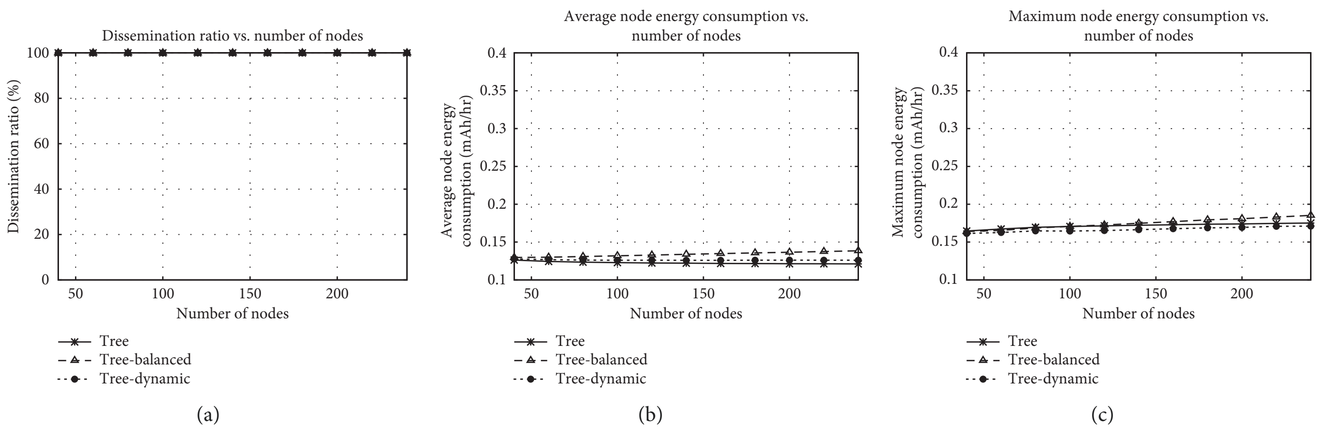


Figure 5: Preliminary experiment results with Tree-based dissemination. In this experiment, number of nodes was varied from 40 to 240. Tx duration was fixed at 1 second. Links were assumed to be reliable. (a) Dissemination ratio. (b) Avg. node energy consumption. (c) Max. node energy consumption.

between the two. Also, while the average energy consumption does not increase for Tree and Tree-Dynamic, the maximum energy consumption of these protocols increases with number of nodes. This means that energy consumption is not well-balanced among nodes, especially when the node density is high. Looking at each of the three variations, Tree-Balanced does not achieve load balancing as expected and performs worse than Tree in terms of average and maximum energy consumption. The reason is because energy consumption does not only depend on number of child nodes, but also number of neighbors in general because of energy consumption due to invalid and valid receptions. Also, when the number of child nodes is balanced, it reduces the chance that a node will become a leaf node without any child node. Tree-Dynamic spends more energy compared to Tree, but has lower maximum energy consumption which means it is balancing energy consumption better than Tree. Still, the gap between Tree and Tree-Dynamic is not significant, compared to the gap between average and maximum energy consumption. Since Tree-Dynamic performs best in terms of maximum energy consumption, we use this protocol for the next experiment, as well as the performance evaluation in Section 5.

In the second experiment, we have varied the link loss ratio from 0 to 50%, for different Tx durations. The Tree-Dynamic protocol is used in this experiment. Since nodes only forward their packets to child nodes in a tree-based dissemination, it is expected that the protocol is vulnerable to lossy links. In order to improve reliability, nodes should increase Tx duration so that a child node who failed to receive the packet could try once more in the next wake-up period. The results are shown in Figure 6. As expected, the dissemination ratio drops drastically as the link loss ratio increases (compared with Flooding-based dissemination shown in Figure 4). Figures 6(b) and 6(c) show that when Tx duration is high, not only the nodes spend more energy due to increased Tx duration, but because more nodes are disseminated into the network. It can be observed that the gap between maximum energy consumption is higher than average energy consumption, which also indicates that some portions of the nodes spend energy faster than other nodes, which is harmful for achieving long network lifetime.

From preliminary experiments, we can conclude that while the Flooding-based protocol achieves high reliability in the face of lossy links, it generally spends too much energy by creating lots of duplicate packets. On the other hand, the tree-based protocol spends less energy, but is highly vulnerable to lossy links and does not balance energy consumption well among nodes. For reliability and tolerance to packet loss, we would like to allow multiple candidate paths for dissemination. Also, in order to reduce unnecessary energy consumption, we want to minimize the number of duplicate packets in the network. The main idea is to make each node forward the packet to a single node in the neighborhood, which has not received the packet yet. Since nodes wake up at different times in an asynchronous duty-cycling network, a node can opportunistically forward the packet to the node which wakes up the earliest and receives the packet. Near the beginning of the dissemination, a node will have many candidate forwarders since not many nodes have received the packet yet. As more nodes receive the packet, the expected wait time of a sender becomes larger, because the number of neighbors who have not received the packet becomes small. However, since the packet is disseminated on a different path each time, wait time of nodes are expected to be balanced among nodes. The implementation of this idea is the proposed protocol, which is described in detail in the next section.

4. Proposed Protocol: Oppo-Flood

4.1. Overview. We call the proposed protocol Oppo-Flood, which reflects the idea of opportunistic forwarding in data dissemination. The main idea of Oppo-Flood is to make each node opportunistically forward packet to a subset (mostly one) of its neighbors so that all the nodes can eventually receive the packet. Consider the scenario in Figure 7. Figure 7(a) shows an example network topology where dotted edges show neighbor relations. A Tree-based dissemination protocol would build a tree structure as in

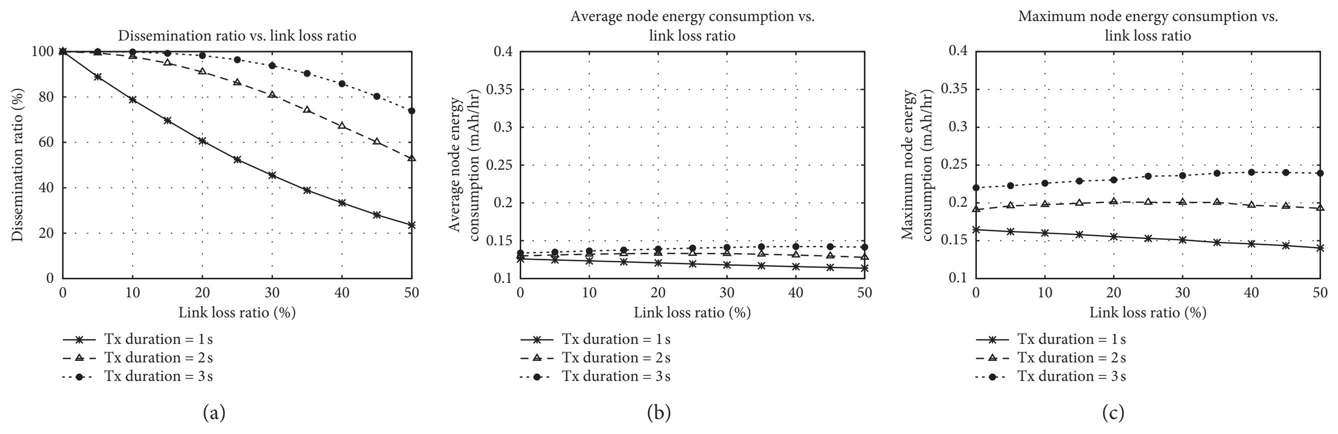


Figure 6: Preliminary experiment results with Tree-based dissemination. In this experiment, link loss ratio was varied from 0 to 50%. The number of nodes was fixed at 100. For Tx duration, 1, 2, and 3 seconds were used. (a) Dissemination ratio. (b) Avg. node energy consumption. (c) Max. node energy consumption.

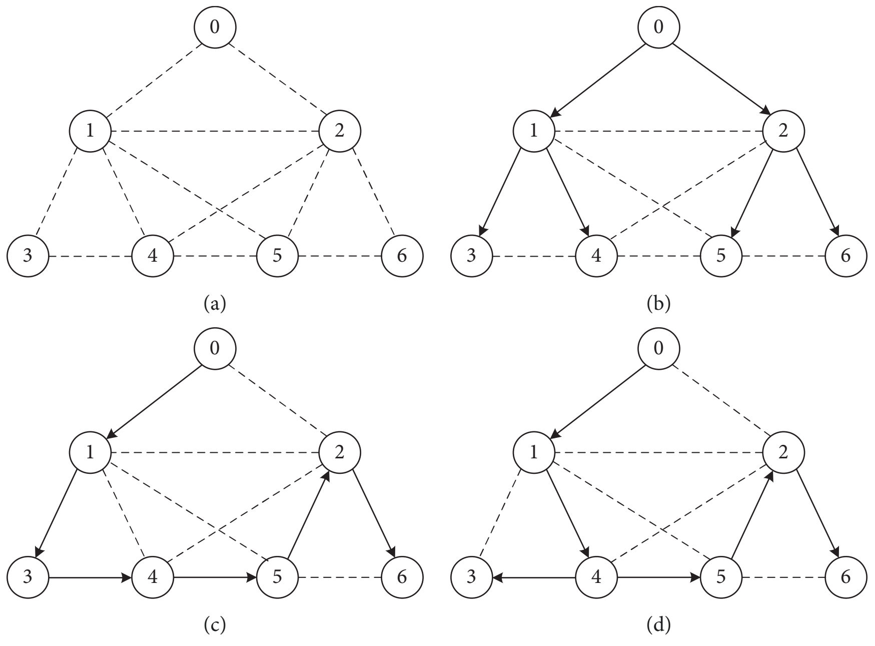


Figure 7: An example network scenario. Node 0 is the source node. (a) The network topology where dotted edge indicates neighbor relations. (b) The paths of Tree-based flooding. (c, d) Example paths of opportunistic flooding. In (c), node 3 wakes up before nodes 4 and 5 to receive packet from node 1. In (d), node 4 wakes up before nodes 3 and 5. (a) Network topology. (b) Tree-based flooding. (c) Opportunistic flooding: case 1. (d) Opportunistic flooding: case 2.

Figure 7(b), and send packets along the tree. In that case, node 1 and node 2 should always forward the packets to their child nodes, and their energy will drain faster than other nodes. Instead, in Oppo-Flood, we let each node transmit its packet to whoever wakes up first among its neighboring nodes. In Figure 7(c), node 0 forwards its packet to node 1 because after node 0 starts transmitting, node 1 wakes up first and receives the packet. On receiving packet, node 1 sends an ACK back to node 0. When node 0 receives the ACK, it stops transmission and goes back to sleep. Node 1 also opportunistically forwards its packet to node 3, who wakes up before nodes 4 and 5. Node 3 then forwards its packet to node 4. Node 1 may wake up before node 4, but since node 1 has already received the packet, it discards the packet and does not send an ACK to node 3. Similarly, node 4 forwards the packet to node 5, node 5 sends its packet to node 2, and finally node 2 sends its packet to node 6. In another case shown in Figure 7(d), when node 1 starts transmitting, node 4 wakes up before nodes 3 and 5 to receive the packet. In this case, if node 4 opportunistically sends its packet to 5 and goes back to sleep, node 3 cannot receive the packet. Thus, node 4 needs to know that it has to forward its packet to both 3 and 5, because otherwise one of the nodes cannot receive the packet.

The benefit of using opportunistic forwarding is that every time a flooding packet is sent from the source node, it can take a different path to reach all the nodes in the network. Therefore, energy consumption is balanced among nodes, without explicitly measuring residual energy and explicitly guiding packets to nodes with higher residual energy. Also, opportunistic forwarding can be beneficial when link conditions are unstable. In Figure 7, suppose the link between node 0 and node 1 is unstable, and so node 1 cannot receive packet from node 0 intermittently. Using a Tree-based forwarding, node 0 has to send its packet to node 1, so node 0 wastes energy trying multiple times to send its packet to node 1. However, in an opportunistic flooding, node 0 can send its packet to node 2 and let other nodes forward the packet to node 1. Node 0 sends its packet to node 2 without explicitly choosing another path, because whoever replies first with an ACK becomes the forwarding node.

The basic idea of opportunistic flooding is simple, but in order to achieve reliability and energy-efficiency, two major challenges need to be addressed. First, as shown in Figure 7(d), a node has to figure out whether it is enough to forward the packet to just one of the neighbors or it should send the packet to multiple nodes. More specifically, if the node is an articulation point which disconnects the graph when removed, it needs to forward the packet to nodes in each of the subgraphs. Second, duplicate copies are inevitably created due to various reasons. For example, when node 0 is transmitting its packet, nodes 1 and 2 may wake up at the same time receiving the packet. If duplicate packets are created, the protocol should minimize energy wasted from sending redundant packets. In order to do that, nodes should keep track of which nodes have received the packet. In the following section, we describe the details of protocol operation.

4.2. Basic Protocol Operation. In Oppo-Flood, each node is assigned a unique identifier. For example, if 128 nodes are deployed, they are assigned IDs from 0 to 127. When a packet is disseminated, the receive status of the packet is included as a bitmap in the packet header. For example, if there are 128 nodes, 128 bits (16 bytes) are required in the header to indicate the receive status of the packet. Since one bit of header space is required for each node in the network, the header size could become too large for a large-scale network. In that case, we can divide the network into multiple groups and assign IDs for each group (plus the group ID). If there are multiple groups, the packet is first delivered to the group leaders, and from thereon, the packet is disseminated in each group using opportunistic flooding. Here we assume that there is only a single group in the network.

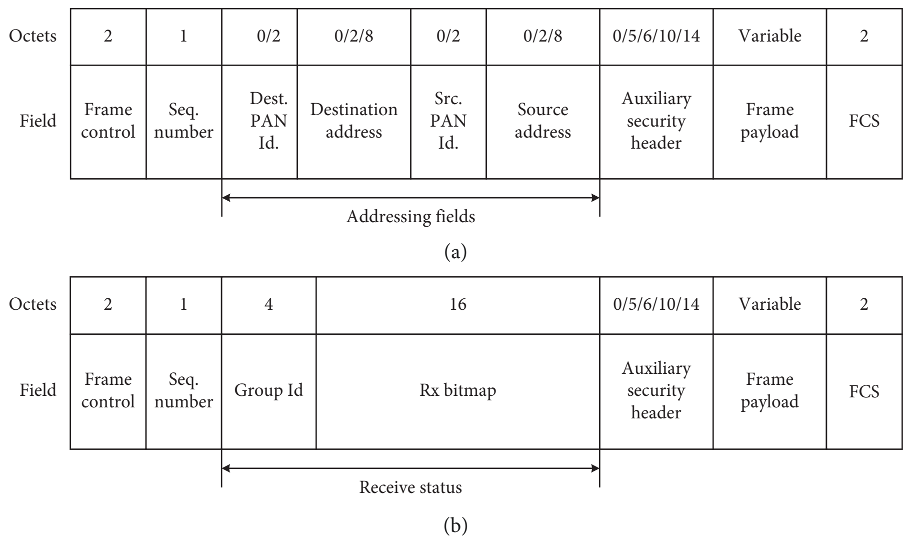
An example packet header of Oppo-Flood is described in Figure 8. Figure 8(a) shows the packet header of IEEE 802.15.4, a standard for low-rate wireless personal area networks (LR-WPANs). The Oppo-Flood replaces the addressing fields in the header with the receive status fields that consist of group ID and the Rx bitmap. The size of the fields could be adjusted according to network scale. Also, the receive status should be included in the ACK frame. Including Rx bitmap in the header could significantly increase the packet size, especially if the payload size is small and the network is large. The increased packet size could cause additional energy consumption for the sender and the receiver. However, the sender wait time is not affected by the packet size, but by wake-up interval of nodes.

In Oppo-Flood, each node maintains a neighbor table, which includes the neighbor nodes of each neighbor. In other words, each node should have knowledge on its two-hop neighbors. There are several ways of obtaining neighbor information. First, after deployment, an initial phase can take place where all nodes are active and nodes periodically broadcast messages to its neighbors. Once a node discovers a neighboring node, it includes the IDs of the neighboring nodes in the message, so that its neighbors can find out the two-hop neighbors. This initial phase is similar to the tree-building phase in tree-based collection or dissemination protocols. Also, after the duty-cycling is started, a node can send request messages to its neighbors so that they can reply with ACK messages. Each node includes its neighbor information in the ACK messages so that the requesting node can acquire two-hop neighborhood information.

Here, we illustrate the operation of Oppo-Flood using an example scenario shown in Figure 9. Figure 9(a) shows a dissemination flow which matches the scenario shown in Figure 7(c). For each edge where the packet is forwarded, the contents of the Rx bitmap in the packet header is shown. First, node 0 starts the dissemination by transmitting its packet. In the packet header, node 0 sets the first bit (which indicates node 0) of the Rx bitmap, because it is the only node that currently has the disseminated packet (it is also possible to omit the source node to save 1 bit). Node 1 receives the packet header and finds out that the packet is a new packet by reading the sequence number field. Also, node 1 reads the Rx bitmap and updates its neighbor table as shown in Figure 9(b).

Now, node 1 needs to decide which node it should forward the packet to, which is decided locally based on the two-hop neighbor information. After receiving the packet, node 1 finds out that nodes 2, 3, 4, and 5 need to receive the disseminated packet. The node checks whether these nodes are connected using the information in the neighbor table. In this case, node 1 can find out that all its neighbors are connected. Then, node 1 can forward the packet to whoever wakes up first and finish transmission. Among the neighbors, node 3 wakes up first and receives the packet. Since node 1 already has the packet, node 3 needs to send the packet to node 4. Node 4 has nodes 1, 2, 3, and 5 as neighbors, but through the Rx bitmap, node 4 knows that nodes 1 and 3 have received the packet. Since nodes 2 and 5 are connected, node 4 can forward the packet to any of the nodes, which happens to be node 5 in this case. Similarly, node 5 needs to send the packet to one of node 2 and node 6, and node 2 receives the packet. Finally, node 2 sends the packet to node 6 and the dissemination process is finished.

Let us consider another example shown in Figure 10. This shows a dissemination flow which matches the scenario shown in Figure 7(d). In this case, node 4 receives the packet transmitted by node 1, because as node 1 is transmitting, node 4 wakes up before other nodes and receives the packet.



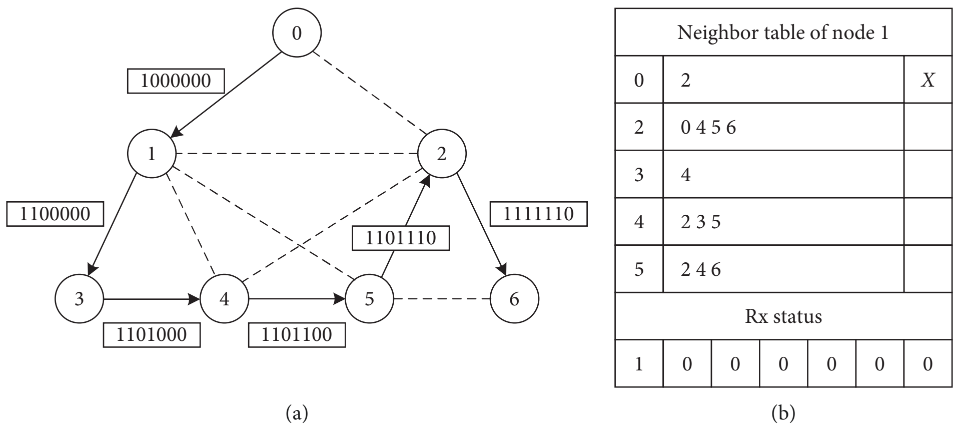
Octets	2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	Variable	2
Field	Frame control	Seq. number	Dest. PAN Id.	Destination address	Src. PAN Id.	Source address	Auxiliary security header	Frame payload	FCS

Addressing fields
(a)

Octets	2	1	4	16	0/5/6/10/14	Variable	2
Field	Frame control	Seq. number	Group Id	Rx bitmap	Auxiliary security header	Frame payload	FCS

Receive status
(b)

Figure 8: The packet header structure of IEEE 802.15.4 and Oppo-Flood. For opportunistic forwarding, the header does not need to include the source and destination address. Instead, Oppo-Flood includes the receive status fields, including group ID and Rx bitmap. (a) IEEE 802.15.4 header. (b) Oppo-Flood header.

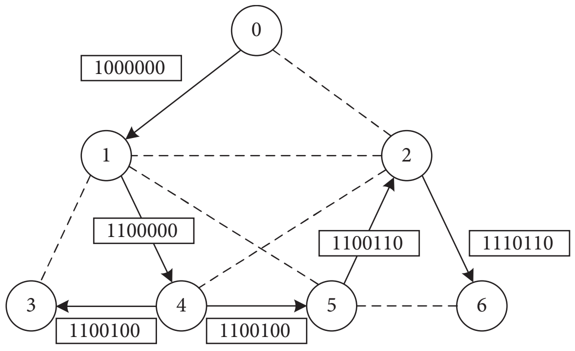


Neighbor table of node 1						
0	2					X
2	0 4 5 6					
3	4					
4	2 3 5					
5	2 4 6					
Rx status						
1	0	0	0	0	0	0

Figure 9: An example operation of Oppo-Flood. (a) The Rx bitmap included in the transmitted packet. (b) The neighbor table of node 1, after the node receives the disseminated packet from node 0. (a) Operation of Oppo-Flood: case 1. (b) Neighbor table of node 1.

Node 4 updates its Rx status, and knows that nodes 2, 3, and 5 need to receive the packet. In this case, although nodes 2 and 5 are connected to each other, node 3 is not connected to either node 2 or node 5. Thus, if node 4 forwards the packet to node 2 or node 5 and finishes transmission, node 3 will not be able to receive the packet. In this case, node 4 forwards the packet to node 3 and one of node 2 and node 5. Generally speaking, a node checks if the neighbors that did not receive the packet form a connected component. If they are all connected, the node can forward its packet to any one of the neighbors and finish transmission. If there are multiple connected components, the node should forward the packet to one node from each of the connected components. If the sender receives ACKs from at least one node from every connected component, it stops transmission and goes back to sleep. Otherwise, the sender continues transmitting the packet until timeout occurs, based on a predefined timeout period.

4.3. Duplicate Packet Handling. In Oppo-Flood, when a node transmits a packet, two or more neighboring nodes can wake-up simultaneously and receive the packet. In that case, since the receivers do not know whether multiple nodes have received the packet, each node forwards the packet independently. The problem is that each copy of the packet does not have the full knowledge of who has received the packet. Consider the scenario shown in Figure 11. When node 0 transmits the packet, node 1 and node 2 wake up at the same time to receive the packet. This is possible because node 1 and node 2 do not coordinate schedules and independently choose their own wake up times. When the two nodes receive the packet, both nodes know that node 0 has the packet but do not know that the packet is received by the other node. Suppose node 1 forwards the packet to node 3 and then the packet goes to node 4. Also, node 2 forwards the packet to node 6 and then to node 5. Now, node 5 knows that the packet is received by nodes 0, 2, 5,



(a)

Neighbor table of node 4						
1	0 2 3 5					*X*
2	0 1 5 6					
3	1					
5	1 2 6					
Rx status						
1	1	0	0	0	0	0

(b)

Figure 10: Another example operation of Oppo-Flood. (a) The Rx bitmap included in the transmitted packet. (b) The neighbor table of node 4, after the node receives the disseminated packet from node 1. (a) Operation of Oppo-Flood: case 2. (b) Neighbor table of node 4.

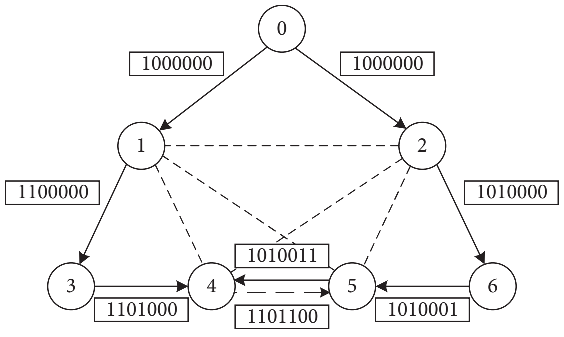


Figure 11: A dissemination scenario where node 1 and node 2 wake up at the same time to receive the packet transmitted by node 0.

and 6 and thinks it should forward the packet to either node 1 or node 4.

When node 5 transmits its packet, any of its neighboring nodes can wake up and receive the packet. In the first case, suppose node 2 wakes up. First, node 2 reads the sequence number field in the packet header and finds out that it is an already received packet. Then, node 2 reads the Rx bitmap field. The Rx bitmap field in the header is "1010011," which means nodes 0, 2, 5, and 6 have received the packet. Node 2 compares the Rx bitmap with its own Rx status, which is "1010000." This means that node 5 already knows all the information that node 2 has. In this case, node 2 aborts reception and goes back to sleep without sending an ACK back to the sender. If node 6 receives the packet, it will show the same behavior.

In the second case, suppose node 1 wakes up and receives the packet. Checking the sequence number field, node 1 finds out that it is a duplicate packet. Then, node 1 compares the Rx bitmap ("1010011") with its Rx status ("1100000"). In this case, node 5 does not know that node 1 has already received the packet. Thus, node 1 sends an ACK to node 5 and includes its Rx status in the packet header. When node 5 receives the ACK, it updates its Rx status using the newly acquired information. The Rx status of node 5 now becomes "1110011." Also, since the Rx status was included in the ACK header, node 5 knows that the packet was already received by node 1. Now, after updating the Rx status, node 5 recalculates the connected components in the neighbor. In this case, node 4 is the only node in the neighborhood that did not receive the packet. So, node 5 continues transmitting the packet with updated Rx bitmap in the packet header.

In the third case, suppose node 4 wakes up and receives the packet. After reading the header and comparing the Rx bitmap ("1010011") with its Rx status ("1101100"), node 4 decides to send an ACK since it has additional information that the sender does not have. When node 5 receives the ACK, it updates its Rx status to "1111111." In this case, node 5 knows that all of its neighbors have received the packet. Thus, node 5 stops transmission and goes back to sleep.

In summary, duplicate packets are inevitably created while disseminating the packets in the network. In order to minimize unnecessary transmission, nodes exchange Rx status included in data and ACK packets in order to gather information from each other. When a node receives a duplicate packet, it checks whether it has additional information in its Rx status that the sender does not have. In this case, the node decides to send an ACK; otherwise, the node does not send an ACK. When the sender receives an ACK that includes an Rx bitmap, it updates its Rx status and decides whether it should continue or stop the transmission.

4.4. Probabilistic Packet Reception. When nodes 1 and 2 wakes up and receive the packet, they will both send ACKs to node 0, which will cause a collision. If their received power is similar, it is possible that node 0 loses both ACKs. Since node 0 did not successfully receive the ACK, it has to continue transmission until it receives an ACK from the forwarder (or the timer expires). Thus, an ACK collision creates duplicate packets in the network, and also it extends the wait time of the sender. If a node has a lot of neighbors that are potential forwarders, its wait time will be short, but the probability of ACK collision will also increase. So there is an optimal number of forwarders which will minimize the network consumption, and this optimal number depends on network parameters such as node density and wake-up interval, as well as environment factors such as link quality.

If a node has too many neighboring nodes that can receive the packet, the high probability of ACK collision will increase energy consumption. In order to control the number of receivers, Oppo-Flood uses probabilistic packet reception. For example, the sender can indicate a probability such as 5/8 using 3 bits in the header. Then, a receiver receives the packet with 5/8 probability by generating a random number. If the node does not receive the packet, it does not send an ACK back to the sender.

The sender chooses the probability in order to meet the expected number of forwarders in the connected component. For example, consider the scenario in Figure 12. Suppose node S is the packet sender, and all the unlabeled nodes have yet to receive the packet. First, node S calculates the connected components of neighbors. Suppose there are two connected components with 6 nodes and 4 nodes in each component. In this case, node S calculates the probability based on the size of the smallest connected components (4 nodes in this example). There is a predefined expected number of forwarders. If the number is 2, then node S chooses the reception probability to be 0.5. This information is included in the packet header, and the neighboring nodes receive packets and send ACKs according to the given probability.

Finding the optimal number of forwarders is out of scope of this paper. In the performance evaluation, we study the impact of expected number of forwarders in metrics such as network lifetime, energy consumption, and dissemination delay.

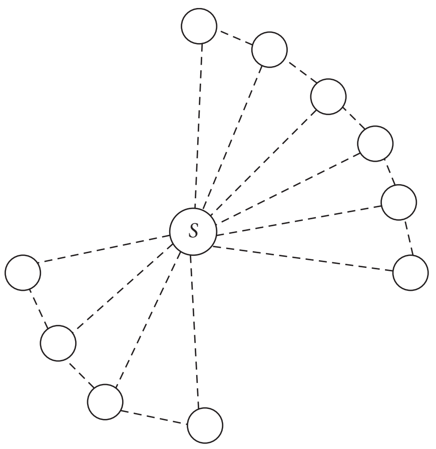


FIGURE 12: An example scenario where node S is sending packets to its neighbors. Node S finds out that its neighbors form two connected components of 6 nodes and 4 nodes. If the expected number of forwarders is 2, node S indicates the reception probability of 0.5, since the smaller component has 4 nodes.

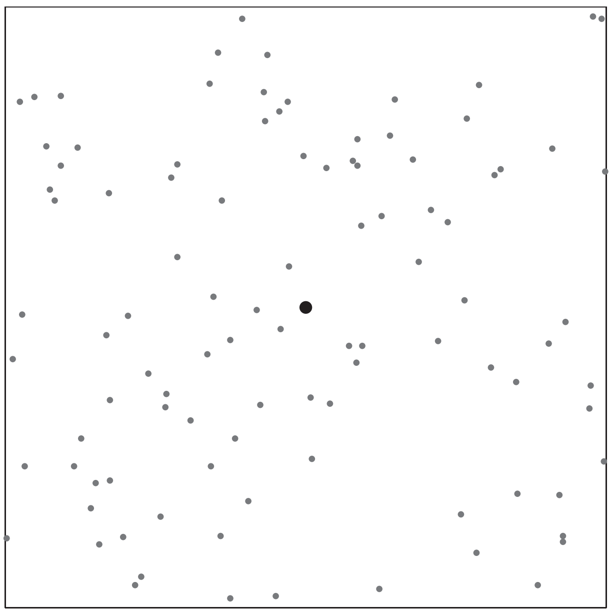


FIGURE 13: An example simulation environment with 100 sensor nodes. The source node (large black dot) is placed at the center of the area.

5. Performance Evaluation

5.1. Simulation Setup. We have evaluated the performance of Oppo-Flood using simulations. The simulator was implemented in Python. At the beginning of a simulation, sensor nodes are randomly deployed in the simulation area, as illustrated in Figure 13. The source node is placed at the center of the area. Then, the nodes start duty-cycling based on independently chosen wake-up times. Otherwise specified, the wake-up interval of every node is 1 second, including the source node. Periodically, the source node transmits a packet to be disseminated to the whole network. The default period of dissemination is 5 minutes.

We use the Log-distance path loss model (equation (1)) to calculate the received signal strength, and successful packet reception is determined based on the signal-to-noise ratio (SNR, equation (2)). In equation (1), the reference path loss (PL_0), the reference distance (d_0), and the path loss exponent (γ) are 46.67 dB, 1 m, and 3, respectively. These values are default values used in the ns-3 simulator [36]. In equation (2), the noise floor is 93.97 dBm, which is also a default value in the ns-3 simulator. The transmit power is 0 dBm for all nodes [37], and the SNR threshold for successful packet reception is 6 dB. With these default parameters, the transmission range is approximately 23.8 m. Additional packet loss is modeled using link loss ratio, which is 0 at default, but we conduct an experiment where we vary the link loss ratio.

$$PL = P_{\text{Tx}} - P_{\text{Rx}} = PL_0 + 10\gamma \log_{10}\frac{d}{d_0}, \quad (1)$$

$$\text{SNR} = \frac{P_{\text{Rx}}}{\Sigma P_I + N_f}. \quad (2)$$

We follow the timing model specified in the BoX-MAC paper [33]. When a node wakes up at its scheduled time, it checks the channel to see if there is any ongoing packet. This is called idle receive check, which requires 5.61 ms. If there is an ongoing packet, the node has to determine if the packet is intended for the node itself and thus it should receive the packet. If the packet is not to be received, the node cancels reception and goes back to sleep. This is called invalid receive, and it requires 20 ms. If the packet should be received, the node receives the packet and replies to the sender by sending an ACK. This is called valid receive, and it takes 50 ms. In the simulations, we divide time into 50 ms slots, and let each node randomly choose one slot for wake up, among the slots in a wake-up interval. For the energy consumption model we follow the CC2420 specification [37]. The current consumption for transmit, listen (or receive), and sleep modes are 17.4 mA, 18.8 mA, and 0.02 μA, respectively. At the beginning of simulation, all nodes are assumed to be equipped with fully charged batteries that have 10,000 mAh capacity.

For evaluation metrics, we use dissemination ratio, network lifetime, average energy consumption of nodes, and dissemination delay. The dissemination ratio indicates how many nodes have successfully received the disseminated packet. The network lifetime is calculated as the time when the first node drains all of its energy [35]. The average consumption of nodes is calculated as the average current consumption of all the nodes per hour (mAh/hr). Finally, the dissemination delay is the duration of time between that when the source node transmits the packet and when the last node in the network receives the packet.

For performance comparison, we have used a Flooding-based protocol and a Tree-based protocol. Although there exist dissemination protocols that use opportunistic forwarding such as [3], the protocol assumes that each node knows the wake-up schedules of neighboring nodes, and so the sender wakes up just before the receiver's wake-up time to send the packet. On the other hand, the proposed protocol uses BoX-MAC where the sender should wait in active mode until the receiver wakes up. Since the underlying assumptions are different, comparing the protocols cannot be fair. Since the Flooding-based approach and the Tree-based approach are the most representative methods in data dissemination and most protocols are their variants, we compare the performance of our protocol to these two protocols.

In the Flooding protocol, the source node transmits the packet for a wake-up duration, so that all of its neighbors wake up and receive the packet. No ACKs are transmitted by the receivers. When a node receives a packet that was not received before, it forwards the packet by transmitting the packet for a whole wake-up interval. If the packet is an already received one, the node simply discards the packet and does not forward it. In the Tree-based protocol, a tree rooted at the source node is established as an initial phase. In order to balance energy consumption, a node periodically selects a parent with the maximum residual energy while in operation (this protocol was called "Tree-Dynamic" in Section 3). When a node receives a packet from its parent node, it sends an ACK back to the receiver. If the packet is not from its parent node, the node discards the packet and does not send an ACK. When a node forwards a packet, the node continues transmitting packet streams until it receives ACKs from all of its child nodes. For the proposed protocol Oppo-Flood, we evaluate two variations of the protocol, "Oppo-Flood-1" and "Oppo-Flood-2." The probabilistic packet reception described in Section 4-D is not used in Oppo-Flood-1, whereas it is used in Oppo-Flood-2. In Oppo-Flood-2, the probability is calculated so that the expected number of forwarders is 6. In the results, each point in the graph is an average of 100 runs with different topology and random number generator seed.

5.2. Results

5.2.1. Varying Number of Nodes. In the first experiment, we have varied the number of nodes from 40 to 240. The nodes were randomly deployed in a simulation area of 100 m × 100 m, and the source node was placed at the center of the area. The result is in Figure 14. Dissemination ratio is not shown since all protocols achieve 100% dissemination ratio regardless of number of nodes. For the network lifetime, the Oppo-Flood protocol achieves longer network lifetime compared to Flooding and Tree-based protocols. For the Flooding and Tree-based protocols, the network lifetime decreases with node density. For the Flooding protocol, every node transmits once for each disseminated packet, for the duration of a wake-up interval. Thus, the energy consumption for transmission is similar regardless of number of nodes. However, more energy is consumed receiving invalid packets, which makes the network lifetime decrease. For the Tree-based protocol, as the number of nodes increases, the nodes near the source node have more child nodes to forward the packet to. Thus, they need to transmit the packet for a longer duration of time in order to receive ACKs from all of its child nodes. Thus, they drain their energy faster and the network lifetime is decreased.

On the other hand, for the Oppo-Flood protocol, the network lifetime increases with number of nodes to a certain extent and then starts decreasing. This is because when the number of nodes is small, nodes have few candidate forwarders and the forwarding path becomes similar to a tree. In this case, nodes near the source node drains energy faster than nodes far away from the source node. As the number of nodes increase, nodes have more candidate forwarders, and especially the nodes near the source node can spend less energy because they can quickly forward to one of the neighbors and finish transmitting the packet. Thus, the energy consumption is well-balanced and the network lifetime becomes longer. However, if the network becomes very dense, the network lifetime starts to drop due to ACK collisions and duplicate packets, as discussed in Section 4-D. The network lifetime of Oppo-Flood-1 drops faster than Oppo-Flood-2, because Oppo-Flood-1 creates more ACK collisions and duplicate packets by allowing more candidate forwarders. Oppo-Flood-2 mitigates the negative effect of duplicate packets by limiting the number of forwarders, and it achieves longer network lifetime than other protocols when the network is very dense. Figure 14(b) shows the average node energy consumption. Here, we can observe that while the energy consumption of the Flooding protocol is significantly higher, the average energy consumption of the other protocols is similar. Also, when the node density is high, the average energy consumption of Oppo-Flood-1 is higher than the Tree-based protocol. Still, the network lifetime of Oppo-Flood-1 is higher than Tree-based. This is due to the fact that energy consumption is more well-balanced in Oppo-Flood-1 compared to the Tree-based protocol. Finally, Figure 14(c) shows the dissemination delay. For the Flooding protocol, the dissemination delay increases up to a certain point and converges afterwards. In the Flooding protocol, each node transmits its packet for a fixed

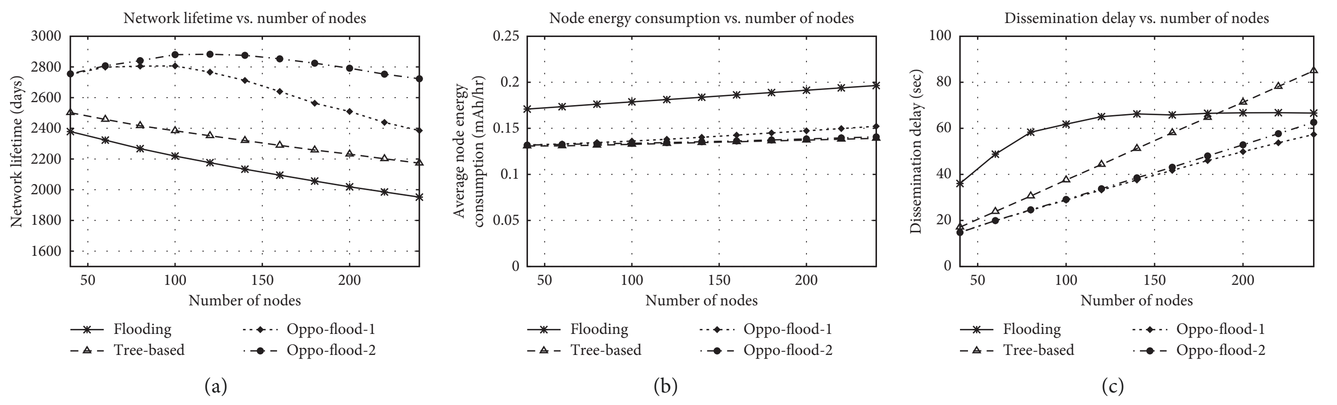


Figure 14: Results varying number of nodes. The number of nodes is varied from 40 to 240, while the area size is fixed to 100 m × 100 m, (a) network lifetime, (b) node energy consumption, (c) dissemination delay.

duration of time. As the number of nodes increases, more nodes need to transmit before all nodes in the network receive the packet. Thus, the dissemination delay increases. However, after a certain point, the number of nodes that cover the whole network does not increase further, although the node density increases. Thus, the dissemination delay converges and does not increase any more. Comparing Tree and Oppo-Flood, we can observe that the Oppo-Flood protocol achieves lower dissemination delay than the Tree-based protocol. This is because the average transmission time of each node is longer for the Tree-based protocol. In the Tree-based protocol, the parent-child relations are not established according to the wake-up time. Thus, a node may need to wait a long time before all of its child nodes wake up and receive the packet. On the other hand, in Oppo-Flood, most of the time, a node finishes transmission after the first neighboring node wakes up and receives the packet. Thus, the average transmission time of each node is shorter than the Tree-based protocol which results in lower dissemination delay. When the node density is very high, the dissemination delay of Oppo-Flood-2 is slightly higher than Oppo-Flood-1, because Oppo-Flood-2 limits the number of neighbors who can receive the packet using probabilistic packet reception. At the cost of increased delay, Oppo-Flood-2 achieves higher network lifetime by reducing number of ACK collisions and duplicate packets.

5.2.2. Varying Area Size. In the next experiment, we have varied the area size from 2500 m^2 (50 m × 50 m) to 250000 m^2 (500 m × 500 m). The number of nodes was fixed at 100. The result is shown in Figure 15. Similar to the previous experiment, dissemination ratio is not shown since all protocols achieve 100% ratio. First, as shown in Figure 15(a), the Oppo-Flood protocol achieves longer network lifetime compared to Flooding and Tree-based protocols. Similar to the previous experiment, the Flooding protocol achieves the shortest network lifetime because all nodes have to transmit for a whole wake-up interval. The Tree-based protocol achieves longer network lifetime than Flooding protocol by reducing the transmission cost, but not by a large margin. For both Flooding and Tree-based protocols, the network lifetime mostly increases when the area size becomes larger. This is mainly because when the area size is large, node density becomes lower and nodes have less number of invalid packets to receive. One exception is that when the area size is very small (2500 m^2), the Tree-based protocol achieves longer network lifetime than when the area size is larger (3600 m^2). When the area size is 2500 m^2, most nodes can be reached by a single hop from the source node. In this case, only the source node transmits the packet for the Tree-based protocol. Thus, no duplicate packets are generated and no energy is consumed due to invalid packet reception. For Oppo-Flood-1 and Oppo-Flood-2, the network lifetime increases with area size up to a certain point but then drops down until it reaches a point. When the area size is very small, large energy is wasted due to ACK collisions and duplicate packets. As the area size increases, the probability that multiple neighbors wake up simultaneously and receive the packet drops, and thus, the network lifetime becomes higher. After a certain point, the effect of ACK collision diminishes. Then, the network lifetime becomes shorter as the area size grows, because nodes have less number of candidate forwarders and so they have to transmit packets for longer durations. When the area size is small, Oppo-Flood-2 achieves higher network lifetime compared to Oppo-Flood-1 by limiting number of neighbors who can receive the packet. This benefit disappears as the area size becomes larger, because the number of neighbors become smaller.

For average node energy consumption shown in Figure 15(b), the Oppo-Flood protocol spends more energy than Tree-based when the area size is small, but the energy consumption becomes similar as the area size becomes larger. It shows that the longer network lifetime achieved by the Oppo-Flood protocol is not by reducing the average energy consumption but by balancing the energy consumption among nodes. For the dissemination delay, the Oppo-Flood protocol achieves smaller delay compared to Flooding and Tree-based protocols. In Figure 15(c), as the area size becomes large, the dissemination delay of Flooding becomes significantly large because more nodes have to transmit in order to cover the entire network. In the Flooding protocol, every node transmits for the duration of a

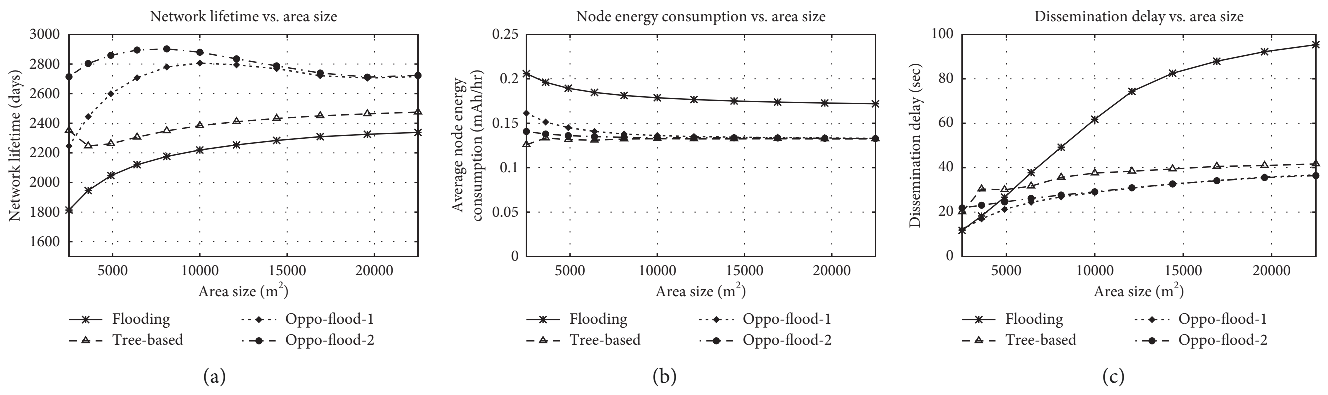


Figure 15: Results varying area size. The area size is varied from 2500 m^2 to 250000 m^2, while the number of nodes is fixed at 100, (a) network lifetime, (b) node energy consumption, (c) dissemination delay.

whole wake-up interval, regardless of whether the neighboring nodes need to receive the packet or not. However, it is possible that all nodes receive the packet when only a small subset of nodes transmit. When the area is small, the number of nodes that can cover the entire network is also small. When the area size becomes larger, the number of nodes needed to cover the whole network becomes larger, and thus the dissemination delay of Flooding becomes larger. In the other protocols, the dissemination delay increases with area size, at a slower rate compared to Flooding. The reason for increase in delay is similar to Flooding, but the Tree-based protocol and the Oppo-Flood protocol limit transmission time of each node using ACKs, which help these protocols to achieve low dissemination delay.

5.2.3. Varying Link Loss Ratio. Until now, whether a node successfully receives a packet is based on the SNR calculated from the Log-distance path loss model. However, channel conditions may vary due to large-scale or small-scale fading, and additional packet losses could occur. In order to study the impact of link quality, we have varied the link loss ratio from 0 to 0.5. For example if link loss ratio is 0.5, a packet is dropped with 50% probability even if the SNR surpasses the threshold. For the Tree-based and Oppo-Flood protocols, the timeout is set to 1 second, which means if the sender does not receive required ACKs within the timeout period, the node finishes transmission and goes back to sleep. The number of nodes is fixed at 100, and the area size is 100 m × 100 m. The result is shown in Figure 16. In this case, we show the dissemination ratio because it is not always 100%. We do not show the dissemination delay, because when the dissemination ratio is very low, dissemination delay is not meaningful because it only measures delay for those who have received the packet successfully.

Figure 16 shows the dissemination ratio of protocols. In Figure 16(a), the Flooding protocol achieves near 100% ratio regardless of link loss, which shows that it is a good candidate for harsh environments and applications in which reliability is more important than network lifetime. In contrast, for the Tree-based protocol, the dissemination ratio drops significantly as the link loss ratio increases. This is due to the unicast behavior of the Tree-based protocol, where a node can only receive the packet from a single parent node. The Oppo-Flood protocol achieves relatively high dissemination ratio in the face of severe link loss. Oppo-Flood-1 achieves 95% dissemination ratio at 50% link loss, while Oppo-Flood-2 achieve 78% at the same level of link quality. Compared to the Tree-based protocol, the Oppo-Flood protocol not only achieves higher dissemination ratio, but also longer network lifetime, as shown in Figure 16(b). In Figure 16(c), we can observe that as the link loss ratio increases, Oppo-Flood starts to spend more energy compared to the Tree-based protocol. Still, the Oppo-Flood protocol achieves longer network lifetime by balancing energy consumption among the nodes. For Oppo-Flood-2, we can observe that the dissemination ratio drops considerably when the link loss ratio is very high. This is because the probability of packet reception does not consider the link quality. A simple improvement is to consider the link quality when calculating the probability of packet reception, but that will require constantly monitoring the link quality, which is not a trivial and cost-free operation.

5.2.4. Varying Wake-Up Interval. For all the previous experiments, the wake-up interval was fixed at 1 second. However, the wake-up interval could be tuned according to application delay requirements. In this experiment, we have varied the wake-up interval from 0.25 to 2.5 seconds. The number of nodes was fixed at 200, and all the other parameters were fixed at their default values. The result is shown in Figure 17.

From Figure 17(a), we can observe that the benefit of Oppo-Flood is increased when the wake-up interval becomes longer (or the duty-cycle becomes lower). When the wake-up interval is short, the network lifetime increases as the wake-up interval becomes longer. Intuitively, the network lifetime becomes shorter if nodes wake-up more frequently. If the wake-up interval is small, the sender wait time will be short, but the benefit is lost by frequent ACK collisions and duplicate packets. As the wake-up interval becomes longer, the network lifetime of Flooding and Tree-based protocols starts to decrease. This is when the cost of

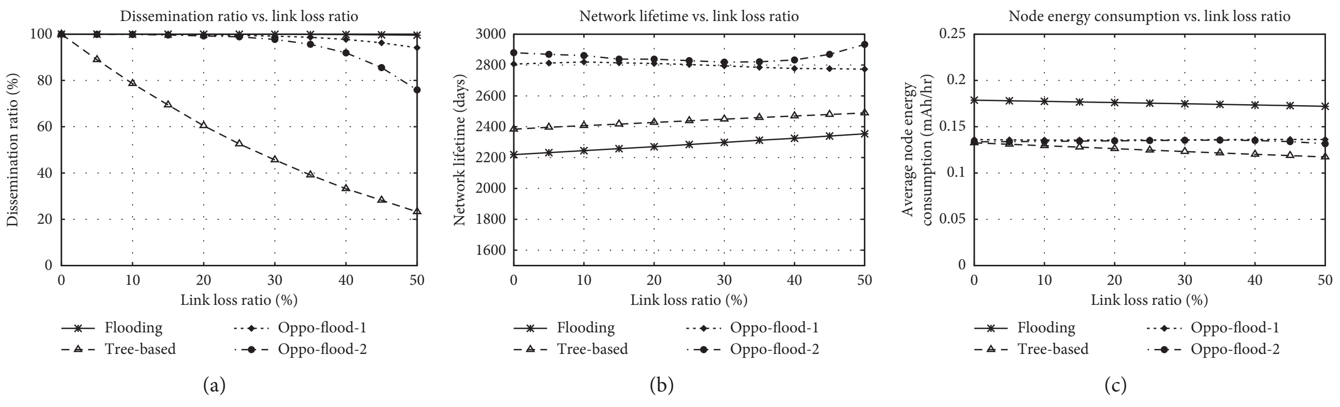


Figure 16: Results varying link loss ratio. The link loss is varied from 0 to 0.5. The number of nodes is 100 and the area size is 100 m × 100 m. (a) Dissemination ratio, (b) network lifetime, (c) node energy consumption.

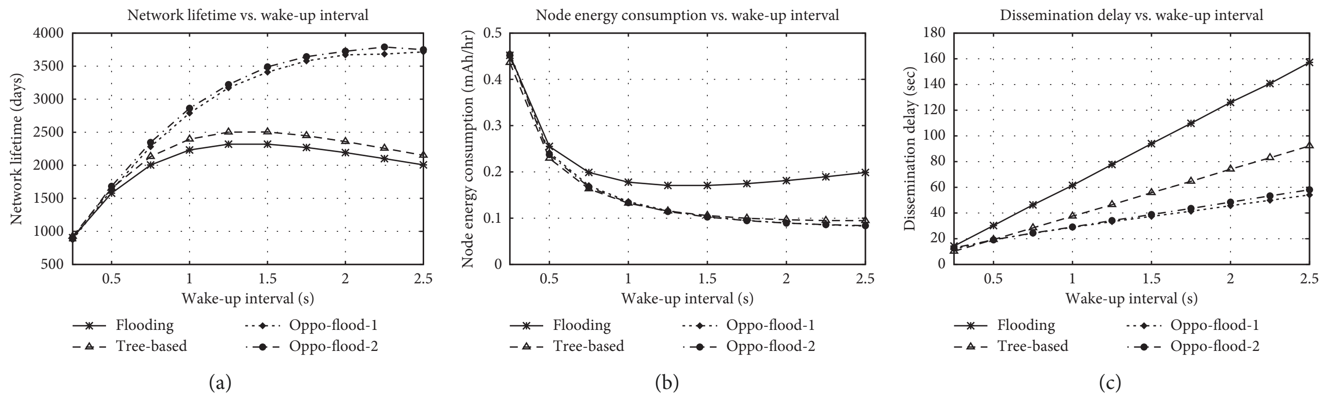


Figure 17: Results varying wake-up interval. The wake-up interval is varied from 0.25 to 2.5 seconds. The number of nodes is 100 and the size is 100 m × 100 m. (a) Dissemination ratio, (b) network lifetime, (c) node energy consumption.

transmission starts to dominate energy consumption more than frequent wake-ups. As the wake-up interval becomes longer, packet senders need to stay in active mode longer in order to forward the packets. However, the network lifetime of Oppo-Flood continues to increase when the wake-up interval is large. This is because Oppo-Flood makes use of opportunistic forwarding to keep the Tx duration low for the packet senders. Similar to previous experiments, Figure 17(b) shows that the average energy consumption of Oppo-Flood is similar to that of the Tree-based protocol, which indicates that the longer network lifetime of Oppo-Flood is achieved through load balancing, by randomly choosing forwarding paths based on when the nodes wake up. On the other hand, the average energy consumption and network lifetime of the Tree-based protocol show that in the Tree-based protocol, a few nodes drain energy much faster than others, thereby shortening the network lifetime. In terms of dissemination delay shown in Figure 17(c), the delay of all protocols increases linearly with the wake-up interval. The Oppo-Flood achieves shorter dissemination delay compared to the Flooding and Tree-based protocols by quickly forwarding packets to a neighboring node who wakes up at the earliest time among all candidate forwarders.

5.2.5. Impact of Probabilistic Packet Reception. In Oppo-Flood-2, we have fixed the expected number of maximum forwarders to 6. The optimal number of forwarders considering the wait time and possibility of ACK collision depends on factors such as wake-up interval and link quality. In this experiment, we have varied the expected number of forwarders from 1 to 30, for wake-up intervals of 1, 2, and 3 seconds. The result is shown in Figures 18(a)–18(c). When the number of forwarders is very small, the network lifetime increases with number of forwarders because more forwarders will reduce wait time of the senders. However, after some point, the network lifetime starts to decrease because the negative effect of ACK collisions and duplicate packets is more significant than the benefit of short wait time. When the wake-up interval is 1 second, the optimal number of forwarders is 5, whereas this optimal number becomes larger when the wake-up interval is longer. This is because when the wake-up interval is longer, the probability of ACK collision is smaller for the same number of forwarders. This result shows that it is necessary to use probabilistic packet reception in order to reduce unnecessary energy consumption and achieve longer network lifetime. For example, when the wake-up interval is 1 second, if we do not limit the

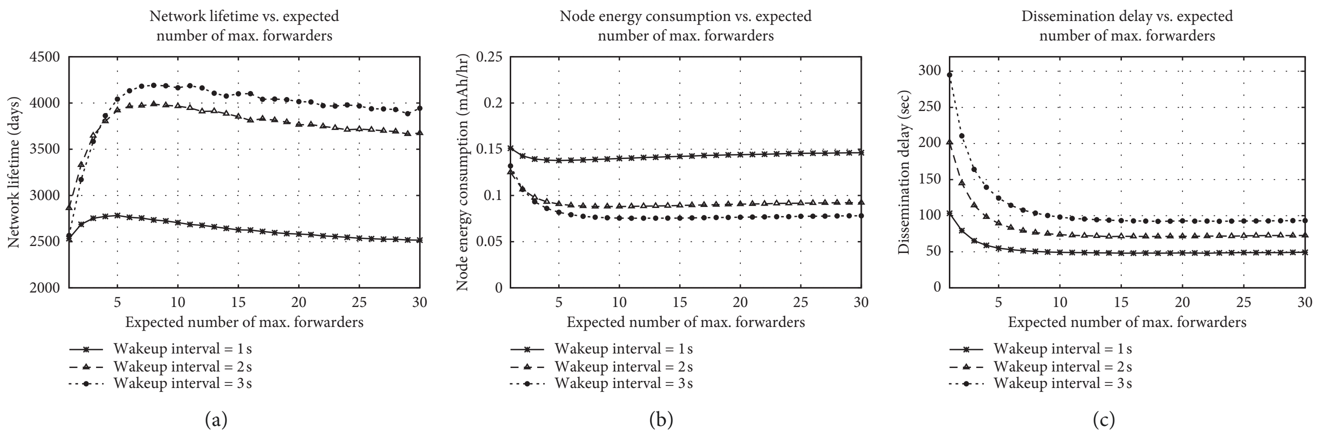


FIGURE 18: Results varying expected number of forwarders. The expected number of forwarders is varied from 1 to 30. The number of nodes is 200 and the are size is 100 m × 100 m. (a) Dissemination ratio, (b) network lifetime, (c) node energy consumption.

number of forwarders, the network lifetime becomes similar to when the number of forwarders is 1.

In summary, the simulation results show that the proposed Oppo-Flood protocol achieves higher network lifetime compared to other protocols such as Flooding and Tree-based protocols. The Oppo-Flood selects dissemination paths opportunistically, thereby reducing the wait time of forwarders and balances energy consumption among nodes by choosing a random path each time a packet is disseminated. The load balancing behavior of Oppo-Flood is shown through network lifetime and average energy consumption, where Oppo-Flood achieves significantly higher network lifetime compared to the Tree-based protocol, while spending comparable amount of energy in average. When the links are lossy, Oppo-Flood achieves higher reliability compared to the Tree-based protocol, which means that in order to achieve a certain level of reliability, Oppo-Flood can use shorter Tx duration compared to the Tree-based protocol, thereby consuming less energy. Oppo-Flood also achieves shorter dissemination delay compared to other protocols by opportunistically forwarding the packet to a neighbor which wakes up at the earliest time.

6. Conclusion and Future Work

In this paper, we proposed an energy-efficient dissemination protocol called Oppo-Flood. Unlike most previous protocols, Oppo-Flood is designed for asynchronous duty-cycling sensor networks where nodes do not know the wake-up schedules of neighbors. In order to overcome the energy-hole problem and balance energy consumption among nodes, the nodes opportunistically forward the packet to a neighbor which wakes up at the earliest time. Using two-hop neighbor information, a node determines if it is an articulation point in the two-hop neighborhood. If the node is an articulation point, it forwards the packet to at least a node in each of the connected components formed by the neighboring nodes. Also, in order to mitigate the negative effects of ACK collision caused by simultaneous wake-ups from neighbors, Oppo-Flood uses probabilistic packet reception based on target number of forwarders. Through extensive simulations, it is shown that the Oppo-Flood protocol achieves higher network lifetime and shorter dissemination delay compared to Flooding and Tree-based protocols in various environments.

The main drawback of Oppo-Flood is that the packet header needs to include a bitmap which increases linearly with number of nodes. When the packet sizes are small, the header size could become excessive overhead, limiting the benefit of the protocol. When the header size becomes large, the sender wait time is not affected, but the time for receiving the packet becomes larger, which increases energy consumption. As briefly discussed in Section 4, for large-scale networks, we can divide the network into multiple groups and apply Oppo-Flood in each of the group. In order to divide the network, we first construct a tree structure rooted at the source node. To balance the number of nodes, each node chooses a parent which has the minimum number of child nodes. After constructing the tree, the first-hop nodes could become the root for each subtree. The dissemination packet is first forwarded to the first-hop nodes, and from thereon, the packets are disseminated using Oppo-Flood. If the tree rooted at a first-hop node is still too large, the tree can be further divided into trees rooted at the second-hop nodes. As a future work, we plan to investigate ways to improve Oppo-Flood, such as efficiently grouping the nodes and reducing the packet header overhead.

Data Availability

The data used to support the finding of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Research Foundation of Korea under Grant NRF-2016R1D1A1B03930289 and NRF-2019R1A2C1005881.

References

[1] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, “Low power, low delay: opportunistic routing meets duty cycling,” in *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Beijing, China, April 2012.

[2] J. So and H. Byun, “Load-balanced opportunistic routing for duty-cycled wireless sensor networks,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 7, pp. 1940–1955, 2017.

[3] S. Guo, L. He, Y. Gu, B. Jiang, and T. He, “Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links,” *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2787–2802, 2014.

[4] Y. Sun, O. Gurewitz, and D. Johnson, “RI-MAC: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks,” in *Proceedings of the ACM Sensor Systems*, Raleigh, NC, USA, November 2008.

[5] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, CA, USA, March 2004.

[6] J. Hui and D. Culler, “The dynamic behavior of a data dissemination protocol for network programming at scale,” in *Proceedings of the ACM Sensor Systems*, pp. 81–94, Baltimore, MD, USA, November 2004.

[7] P. Kyasanur, R. Choudhury, and I. Gupta, “Smart gossip: an adaptive gossip-based broadcasting service for sensor networks,” in *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, Vancouver, BC, Canada, October 2006.

[8] F. Stann, J. Heidemann, R. Shroff, and M. Murtaza, “RBP: robust broadcast propagation in wireless networks,” in *Proceedings of the ACM Sensor Systems*, pp. 85–98, Boulder, CO, USA, November 2006.

[9] L. Huang and S. Setia, “CORD: energy-efficient reliable bulk data dissemination in sensor networks,” in *Proceedings of the IEEE INFOCOM*, Phoenix, AZ, USA, April 2008.

[10] V. Naik, A. Arora, P. Sinha, and H. Zhang, “Sprinkler: a reliable and energy efficient data dissemination service for extreme scale wireless networks of embedded devices,” *IEEE Transactions on Mobile Computing*, vol. 6, no. 7, pp. 777–789, 2007.

[11] T. Zhu, Z. Zhong, T. He, and Z. Zhang, “Exploring link correlation for efficient flooding in wireless sensor networks,” in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, San Jose, CA, USA, April 2010.

[12] W. Dong, Y. Liu, C. Wang, X. Liu, C. Chen, and J. Bu, “Link quality aware code dissemination in wireless sensor networks,” in *Proceedings of the IEEE International Conference on Network Protocols*, pp. 89–98, Vancouver, BC, Canada, October 2011.

[13] M. Miller, C. Sengul, and I. Gupta, “Exploring the energy-latency trade-off for broadcasts in energy-saving sensor networks,” in *Proceedings of the IEEE International Conference on Distributed Computing Systems*, Baltimore, MD, USA, June 2005.

[14] J. Lu and K. Whitehouse, “Flash flooding: exploiting the capture effect for rapid flooding in wireless sensor networks,” in *Proceedings of the IEEE INFOCOM*, Rio De Janeiro, Brazil, April 2009.

[15] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, “Efficient network flooding and time synchronization with glossy,” in *Proceedings of the ACM/IEEE IPSN*, Chicago, IL, USA, April 2011.

[16] M. Doddavenkatappa, M. Chan, and B. Leong, “Splash: fast data dissemination with constructive interference in wireless sensor networks,” in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, Lombard, IL, USA, April 2013.

[17] W. Du, J. Liando, H. Zhang, and M. Li, “When pipelines meet fountain: fast data dissemination in wireless sensor networks,” in *Proceedings of the ACM Sensor Systems*, pp. 365–378, Seoul, South Korea, November 2015.

[18] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. Johnson, “ADB: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks,” in *Proceedings of the ACM Sensor Systems*, pp. 43–56, Berkeley, CA, USA, November 2009.

[19] S. Lai and B. Ravindran, “On multihop broadcast over adaptively duty-cycled wireless sensor networks,” in *Proceedings of the IEEE Distributed Computing in Sensor Systems*, pp. 158–171, Santa Barbara, CA, USA, June 2010.

[20] J. Hong, J. Cao, W. Li, S. Lu, and D. Chen, “Minimum-transmission broadcast in uncoordinated duty-cycled wireless ad hoc networks,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 307–318, 2010.

[21] B. Tang, B. Ye, J. Hong, K. You, and S. Lu, “Distributed low redundancy broadcast for uncoordinated duty-cycled WANETs,” in *Proceedings of the IEEE Globecom*, Houston, TX, USA, December 2011.

[22] T. Duc, D. Le, V. Zalyubovskiy, D. Kim, and H. Choo, “Level-based approach for minimum-transmission broadcast in duty-cycled wireless sensor networks,” *Pervasive and Mobile Computing*, vol. 27, pp. 116–132, 2016.

[23] S. Guo, S. Kim, T. Zhu, Y. Gu, and T. He, “Correlated flooding in low-duty-cycle wireless sensor networks,” in *Proceedings of the IEEE International Conference on Network Protocols*, Vancouver, BC, Canada, October 2011.

[24] Z. Zhao, W. Dong, J. Bu, Y. Gu, and C. Chen, “Link-correlation-aware data dissemination in wireless sensor networks,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 9, pp. 5747–5757, 2015.

[25] F. Wang and J. Liu, “On reliable broadcast in low duty-cycle wireless sensor networks,” *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, pp. 767–779, 2011.

[26] L. Xu, J. Cao, S. Lin, H. Dai, X. Wu, and G. Chen, “Energy-efficient broadcast scheduling with minimum latency for low-duty-cycle wireless sensor networks,” in *Proceedings of the IEEE Mobile Ad hoc and Sensor Systems*, Philadelphia, PA, USA, October 2013.

[27] J. Niu, L. Cheng, Y. Gu, J. Jun, and Q. Zhang, “Minimum-delay and energy-efficient flooding tree in asynchronous low-duty-cycle wireless sensor networks,” in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, April 2013.

[28] K. Han, J. Luo, L. Xiang, M. Xiao, and L. Huang, “Achieving energy efficiency and reliability for data dissemination in duty-cycled wsns,” *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1041–1052, 2015.

[29] F. Sutton, B. Buchli, J. Beutel, and L. Thiele, “Zippy: on-demand network flooding,” in *Proceedings of the ACM Sensor Systems*, pp. 45–58, Portland, OR, USA, June 2015.

[30] L. Xu, X. Zhu, H. Dai, X. Wu, and G. Chen, “Towards energy-fairness for broadcast scheduling with minimum delay in low-duty-cycle sensor networks,” *Computer Communications*, vol. 75, pp. 81–96, 2016.

[31] Z. Cao, D. Liu, J. Wang, and X. Zheng, “Chase: taming concurrent broadcast for flooding in asynchronous duty cycle networks,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2872–2885, 2017.

[32] X. Wu, G. Chen, and S. K. Das, “Avoiding energy holes in wireless sensor networks with nonuniform node distribution,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 710–720, 2008.

[33] D. Moss and P. Levis, “Box-macs: exploiting physical and link layer boundaries in low-power networking,” Tech. Rep. 08-00, Stanford University, Stanford, CA, USA, 2008.

[34] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, “The broadcast problem in a mobile ad hoc network,” in *Proceedings of the ACM MobiCom*, Seattle, WA, USA, August 1999.

[35] I. Dietrich and F. Dressler, “On the lifetime of wireless sensor networks,” *ACM Transactions on Sensor Networks*, vol. 5, no. 1, pp. 1–39, 2009.

[36] T. Henderson, M. Lacage, and G. Riley, “Network simulations with the ns-3 simulator,” in *Proceedings of the ACM SIGCOMM*, Seattle, WA, USA, August 2008.

[37] Chipcon AS, SmartRF CC2420 Preliminary Datasheet, 2004.