OppCode: Correlated Opportunistic Coding for Energy-Efficient Flooding in Wireless Sensor Networks

Xingfa Shen, *Member, IEEE*, Yueshen Chen, Yinqun Zhang, Jianhui Zhang, Quanbo Ge, *Member, IEEE*, Guojun Dai, *Member, IEEE*, and Tian He, *Member, IEEE*

***Abstract*—Existing work on flooding in wireless sensor networks (WSNs) mainly focuses on single-packet problem, while the work on sequential multipacket problem is surprisingly little. This paper proposes *OppCode*, a new opportunistic network-coding-based flooding architecture for multipacket dissemination in WSNs, where both unreliable and correlated links commonly exist. Instead of flooding a single packet each time, each node encodes multiple native packets chosen from a specific fixed-size page to an encoded packet and then rebroadcasts it further. The key idea consists of two parts. One is *opportunistically coding decision*, in which each node grasps every possible coding opportunity greedily to maximize its aggregate coding gain of all neighbors based on the probabilistic estimation of packets each neighbor already has. The other is *paged collective acknowledgements* (ACKs), in which one rebroadcast acts as not only an implicit ACK of successful disseminations of all packets in the entire page for the sender, but also probabilistic ACK to update page-scale per-packet coverage estimations for its neighbors in a batch. Experiments based on extensive simulations and 21-node testbed show that OppCode significantly increases performance of multipacket flooding in terms of reliability, transmission overhead, delay, and load balance.**

***Index Terms*—Link correlation, multipacket flooding, network coding, wireless sensor networks (WSNs).**

I. Introduction

In WIRELESS sensor networks (WSNs), flooding is a fundamental communication primitive supporting many important high-level protocols and applications, such as data dissemination [1], time synchronization [2], key management [3], and multihop routing [4]–[6]. An implicit assumption in traditional flooding mechanisms [7], [8] is that the underlying wireless links are independent of each other. However, recently, the existence of link correlation in wireless networks is confirmed and explored by Srinivasan *et al.* [9], Zhu *et al.* [10], and Wang *et al.* [11], which can be utilized for further performance improvement of flooding protocols. Having deep insights into this phenomenon, Zhu *et al.* [10] successfully reduces the number of reception acknowledgements (ACKs) by employing a mechanism named *collective ACKs* in which one sender can infer the success of a transmission to its receiver based on the ACKs from other neighboring receivers.

Although most existing flooding solutions are aiming at addressing single-packet dissemination, lots of applications of WSNs often have the need of flooding *multiple* sequential packets to complete a specific mission. For example, in over-air reprogramming applications, the program images to be disseminated by flooding protocol may reach sizes up to 128 kB [1], which must be fragmented to numerous small packets due to the fact that the default maximum size of message payload in TinyOS is only 29 B.

In WSNs with unreliable and correlated links, applying single-packet flooding algorithms directly to handle multiple-packet flooding problems will cause substantial performance degradation due to the following factors.

1) Diversity of packet loss patterns: As loss patterns vary across different next-hop nodes, each next-hop node may lose different packets. The sender must keep sending missed packets of every next-hop node until every next-hop node receives all the packets in a page, which will consequently increase contention and collisions in the network.
2) Redundant ACKs: In either per-node ACKs [7], [8] or collective ACKs [10], a sender cannot use the ACK for a specific packet to estimate the reception of other packets sent from the same sender if no inter-packet dependance is assumed.
3) Increased delay: Usually, to decrease contention, single-packet flooding algorithm triggers consecutive packet transmissions at traffic-adaptive time intervals. This will lead to significant increase in end-to-end data dissemination delay.

In this paper, we present a comprehensive study on multipacket flooding problem in the context of WSNs with unreliable and correlated links, and first propose OppCode, a new opportunistic network-coding-based multipacket flooding algorithm.

Manuscript received December 09, 2014; revised April 18, 2015; accepted May 05, 2015. Date of publication May 20, 2015; date of current version December 02, 2015. This work was supported in part by the National Science Foundation of China (NSFC) under Grant 61190113, Grant 61473109, Grant 61172133, and Grant 61202093, and in part by the Zhejiang Provincial Natural Science Foundation under Grant LY14F020047 and Grant LY14F020042. Paper no. TII-14-1359. *(Corresponding author: Guojun Dai.)*

X. Shen, Y. Chen, Y. Zhang, J. Zhang, and G. Dai are with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: shenxf@hdu.edu.cn; chenys4@vip.qq.com; zhangyinqun007@qq.com; jh_zhang@hdu.edu.cn; daigj@hdu.edu.cn).

Q. Ge is with the School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: qbge@hdu.edu.cn).

T. He is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: tianhe@cs.umn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2015.2436336

The driving idea behind our design is network coding, which allows the relay nodes to mix the information content in the packets before forwarding them.

The opportunistic coding-based flooding design consists of the following enabling components: the first is *opportunistic listening*, in which each node estimates its neighbors' receptions of flooding packets based on the condition of link quality and correlation. The second is *coding decision*, in which a node can decide whether it has an opportunity for coding or not based on the reception estimation of its neighbors. If the node get the opportunity, it encodes (i.e., XOR) certain packets and sends the encoded packet. To measure coding opportunity, we propose a metric called total (or *aggregate*) coding gain. Then, a greedy coding decision algorithm is proposed. The third is *coverage updating*, in which the sender and receivers update the coverage distribution across its neighbors after a packet is sent or received, respectively.

The key novelty of this work lies in the *opportunistically coding decision-making*, in which each node locally grasps any possible coding opportunity to encode packets for rebroadcasting based on the estimation of the packets each neighbor already has, so that the aggregate coding gain of all neighbors is maximized while the transmission overhead and dissemination delay are kept as low as possible. Specifically, the major contributions of this work are as follows.

1) To the best of our knowledge, this is the first comprehensive study of opportunistic multiple-packet flooding problem in WSNs with unreliable links and high-link correlations.
2) We propose a new opportunistic network-coding-based multipacket flooding architecture, named OppCode, based on per-packet coverage estimation.
3) A greedy coding decision algorithm is proposed to maximize the aggregate coding gain.

This paper is organized as follows. Section II discusses the related work. The motivation of this paper is discussed in Section III. Then, Section IV introduces the key mechanisms of OppCode. Section V describes our main protocol design, followed by its evaluation in Sections VI and VII. Section VIII concludes this paper.

II. Related Work

Flooding plays important roles in most applications of WSNs [12]–[16], which supports kinds of high-level protocols and services [1], [2], [4], [17]–[24]. However, most existing flooding protocols (e.g., DCB [7], RBP [8], and collective flooding (CF) [10]) only focus on addressing single-packet flooding problem in essence, where multipacket flooding tasks are treated as multiple *independent* single-packet subtasks. In this paper, our work aims at handling sequential multiple-packet flooding problem utilizing inter-packet dependance relationship in a specific page.

This work uses network coding as its core idea aiming at lowering energy cost, in which the sender mixes multiple packets before rebroadcasting instead of flooding single packet. Network coding has been proven that it has the capability to improve network throughput and energy efficiency. A pioneering work [25] by Ahlswede *et al.* has demonstrated that the fact mixing information from different flows in intermediate nodes in the network can achieve the broadcast capacity, and many recent papers follow this idea and extend it to other aspects of networking.

Due to the broadcast nature of wireless networks, network coding has been adopted to support various protocols in wireless networks, e.g., COPE [26] and UFlood [27], and achieved vast performance gains by permitting intermediate nodes to carry out algebraic operations on the incoming data. A concept of opportunistic coding is first introduced by COPE [26], which is designed for *unicast* traffic in some specified wireless environment. We try to extend the opportunistic coding approach to *broadcast* communication pattern in WSNs.

Another motivation of the proposed OppCode protocol [28] is the observation of link correlation generally existing in wireless networks, which is deeply explored by Srinivasan *et al.* [9], Zhu *et al.* [10], and Wang *et al.* [11]. Compared with CF [10], our work combines network coding with link correlation to handling multipacket flooding problem, which aims at achieving the goal of both energy efficiency and network reliability.

Work [29] has similar idea with ours that applies network coding to WSNs with link correlation. It shows the potential of link correlation and network coding-based solutions for data dissemination in WSNs, which adopts *random linear coding* methods that need to solve linear equations for decoding native packets and thus has high computation cost compared with XOR coding that we employed. Moreover, when a node receives an encoded packet that is not innovative, this packet is useless that makes no contribution to decoding. This feature leads to additional energy waste.

Work [30] studies network coding and link correlation in WSNs from another point of view, which analyzes the impact of link correlation on network coding and builds a general model for both unicast and broadcast protocols. While in OppCode, we propose a detailed page-based multipacket coding mechanism under the existence of link correlation. Though both making coding based on link correlation aiming at transmission efficiency, [30] and our work employ totally different coding strategies designed for different networking models and applications.

III. Motivation

In this section, we first demonstrate the existence of link correlation, then we illustrate the benefit of network coding on broadcast protocols.

A. Existence of Link Correlation

Link correlation has been studied in [9] and [10] showing that when a sender sends out a broadcasting packet, the receptions at its receivers are not independent of each other. To verify this statement, an indoor experiment was conducted. In this experiment, 29 TelosB nodes are deployed to form a star topology. Sender is the central node and the others act as receivers. The sender broadcasts a packet in every 150 ms, and every packet is identified by a sequence ID. The total number

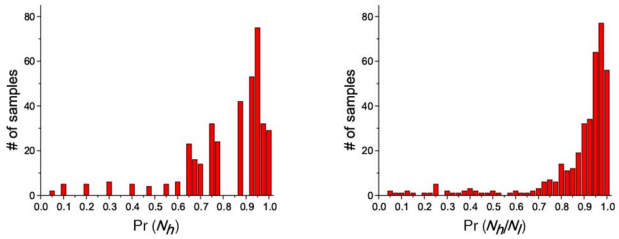


Fig. 1. Statistics of receiving probability.

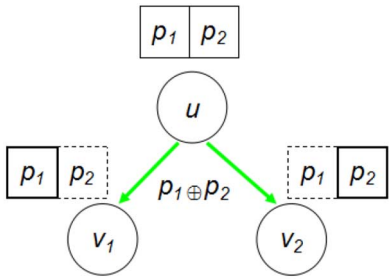


Fig. 2. Benefit of network coding.

of packets broadcasted is 2000. For each pair of receivers, we counted the number of packets received by nodes with higher link quality (denoted as N_h) when nodes with lower link quality (denoted as N_l) have received the same packet successfully, i.e., $P_r(N_h|N_l)$. We compare $P_r(N_h|N_l)$ with the number of successful receptions at N_h regardless of N_l, i.e., $P_r(N_h)$. We found that for about 86% of all the receiver pairs, the former is greater than the latter, i.e., $P_r(N_h|N_l) \geq P_r(N_h)$. From the distribution shown in Fig. 1, we can clearly see that the conditional probability $P_r(N_h|N_l)$ is closer to 1 than $P_r(N_h)$, which verifies the existence of link correlation.

B. Benefit of Network Coding

Network coding has been widely used in broadcast protocols as it has great potential to improve the performance of broadcast applications by mixing multiple packets in the intermediate nodes. Fig. 2 shows the benefit of network coding on broadcast protocols, in which a block with solid borderline means a received packet and a block with dashed borderline means a lost packet. The receivers v_1 and v_2 received packet p_1 and p_2, respectively, and also needed one packet p_2 and p_1, respectively. In traditional ways, the sender u needs two transmissions to make sure its receivers receive packet p_1 and p_2 if the link quality is assumed 100%. With the help of network coding, the number of packets transmitted from sender u can be reduced from 2 to 1 by broadcasting a XORed packet $p_1 \oplus p_2$.

From the above example, we see that the nature behind network coding is that the forwarder encodes the native packets and broadcasts them with a coded packet using one transmission, instead of sending multiple packets one by one.

IV. Key Mechanisms

The main objective of OppCode is to reduce the total number of transmissions while providing reliable multipacket

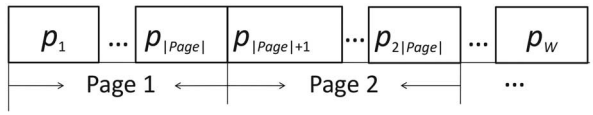


Fig. 3. Example of page division.

dissemination. In OppCode, sequential packets to be disseminated are grouped into fixed-size pages. A node is called *page-covered* (for short, *covered* is used in the rest of this paper) when it receives all the packets of a specific page. Covered nodes are responsible for rebroadcasting this page further to uncovered nodes in the network. There are two key mechanisms in OppCode design as follows.

1) Opportunistic network coding: Network coding is conducted opportunistically in a specific page using a greedy coding decision method before the encoded packet is sent.
2) Paged collective ACKs considering link correlation: A flooding packet overheard or received serves as both a direct and paged collective ACKs for the sender and its neighbors, respectively.

The mechanisms of OppCode are based on per page, which are different from traditional per-packet-based ACKs. For a flooding task that needs to broadcast multiple packets, the first step is to divide a packet stream into few pages with fixed length. Assuming that the stream contains W packets and the page size is $|\text{Page}|$, we can get $\lceil \frac{W}{|\text{Page}|} \rceil$ pages, as shown in Fig. 3.

It is worth noting that page is the basic processing unit not only for collective ACKs but also for opportunistic coding. The page size directly influences the performance of OppCode scheme, i.e., if one page contains too many packets, the cost for network coding decision is quite large. So, it is obvious that the page size should be limited to some reasonable range to avoid high computation overhead. We will evaluate the performance of OppCode while varying the page size based on extensive simulation-based experiments.

A. Opportunistic Network Coding

In OppCode, network coding is opportunistically determined based on a node's estimation about its neighbors, where each node learns its neighbors' *per-packet coverage* in a specific page, and then a sender makes coding decision when detecting any coding opportunity. In this section, we first present a simple example to illustrate this idea in Section IV-A1 and then describe the detailed design of coding decision scheme in Section IV-A2.

1) Conceptual Example of Coding Decision Making: Network coding allows the intermediate node to mix content in the packet before forwarding it, and its aim is to maximize the throughout. The core of network coding is to make a good coding decision, which directly influences the improvement in the performance of network coding. The following simple example gives a brief illustration about when and how to make network coding. Apparently, the coding decision should be made based on the principle that the selected coding option should make more receivers of an encoded packet able to decode

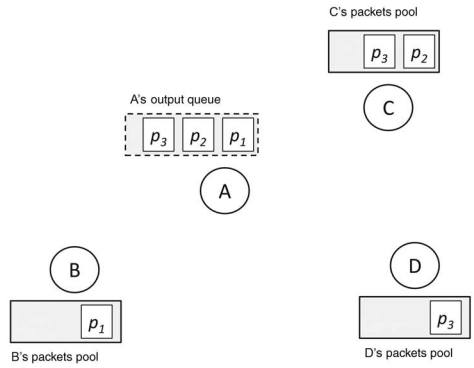


Fig. 4. Example of coding decision-making.

TABLE I
GOOD OR NOT OF DIFFERENT CODING OPTIONS

Coding option	Good or not?	Which nodes can decode?
$p_2 \oplus p_3$	Bad	D
$p_1 \oplus p_2 \oplus p_3$	Bad	C
$p_1 \oplus p_2$	Better	B, C
$p_1 \oplus p_3$	Best	B, C, D

native packets, whose coding benefit is maximized. As shown in Fig. 4, node A has a three-packet page p_1, p_2, p_3 to be disseminated. Its neighbors perhaps have already been covered by some of these packets. Assume that A knows what packets each neighbor has, just as shown in Fig. 4. Several coding options for A have been listed in Table I. First, A could send $p_2 \oplus p_3$. Since D has p_3 in store, it could be XOR p_3 with $p_2 \oplus p_3$ to decode a native packet p_2. However, B does not have p_2 or p_3, so it cannot decode the XOR-ed packet. Thus, sending $p_2 \oplus p_3$ would be a bad coding decision for A, as only one neighbor can benefit from this transmission. The second option in Table I is also bad, because only C can decode a native packet p_1. The third option shows a better coding decision for A. Sending $p_1 \oplus p_2$ would allow both neighbors B and C to obtain one native packet from a single transmission. Therefore, the best coding decision for A would be $p_1 \oplus p_3$, which would allow all three neighbors to decode their respective packets all at once.

2) Aggregate-Coding-Gain-Based Coding Decision: We first present the aggregate-coding-gain-based coding decision method, a detailed solution to make coding decisions in OppCode, based on a metric named aggregate coding gain. This metric is used to measure the effectiveness of a coding option. Higher the value of this metric is, more effective the coding option is. The coding option with highest gain is chosen to be used for rebroadcasting. Specifically, aggregate coding gain is defined as the sum of expectation of number of native packets that each neighbor of the sender can decode successfully as follows:

$$\mathrm{Gain}_u(\Omega) = \sum_{k \in N(u)} \mathrm{Gain}_u^k(\Omega). \tag{1}$$

Here, $\mathrm{Gain}_u(\Omega)$ represents the sender u's coding gain for a certain coding option Ω. $N(u)$ is u's neighbor set. $\mathrm{Gain}_u^k(\Omega)$ indicates u's expectation of the quantity of native packets that k can successfully decode with the option Ω.

$\mathrm{Gain}_u^k(\Omega)$ can be computed using mathematical expectation formula as follows:

$$\mathrm{Gain}_u^k(\Omega) = 1 \cdot \mathrm{Pr}^k + 0 \cdot (1 - \mathrm{Pr}^k) \tag{2}$$

where Pr^k indicates the probability that k can receive and decode successfully. Obviously, Pr^k can be computed as follows:

$$\mathrm{Pr}^k = l(u,k) \cdot \mathrm{Pr}_{\mathrm{decode}}^k(\Omega). \tag{3}$$

In the above equation, item $l(u,k)$ represents the link quality from u to k, while $\mathrm{Pr}_{\mathrm{decode}}^k(\Omega)$ indicates that k successfully decodes a native packet under the condition that the encoded packet with option Ω is received successfully. So, the aggregate coding gain can be computed by

$$\mathrm{Gain}_u(\Omega) = \sum_{k \in N(u)} l(u,k) \cdot \mathrm{Pr}_{\mathrm{decode}}^k(\Omega). \tag{4}$$

The next essential step is to compute $\mathrm{Pr}_{\mathrm{decode}}^k(\Omega)$, which depends on two factors: 1) the coding option Ω; and 2) node ks per-packet coverage status for the specific page. Obviously, k can decode a specific native packet p_i only when the received packet is encoded with a native packet set $\mathrm{natPktSet}(\Omega)$ and k has already received all the packets in this set except p_i before current transmission. So, $\mathrm{Pr}_{\mathrm{decode}}^k(\Omega)(i)$ is the probability that node k just already has all the packets in the set $\mathrm{natPktSet}(\Omega) - p_i$.

Fig. 5 shows a more specific example illustrating how aggregate coding gain is computed. For a page with three packets p_1, p_2, p_3, the sender u has maintained its neighbor' per-packet coverage probabilities $\mathrm{Cov}_u^{N_j}(i)$ of its neighbor N_j from u's perspective, as shown in Fig. 5. Node u has a few coding options, i.e., p_1, p_2, p_3, $p_1 \oplus p_2$, $p_1 \oplus p_3$, $p_2 \oplus p_3$, and $p_1 \oplus p_2 \oplus p_3$. Without loss of generality, we take option $p_1 \oplus p_2$ for instance. The key lies in the computation of $\mathrm{Pr}_{\mathrm{decode}}^{N_j}(p_1 \oplus p_2)$. If N_j has packet p_1 or p_2 in store before the current transmission, then it can decode one packet. Otherwise, if N_j has both p_1 and p_2 in store, or N_j does not have either packet p_1 or p_2, N_j can decode none. From the above analysis, we can easily get $\mathrm{Pr}_{\mathrm{decode}}^{N_j}(p_1 \oplus p_2)$ $(j = 1, 2)$ as follows:

$$\begin{aligned}\mathrm{Pr}_{\mathrm{decode}}^{N_1}(p_1 \oplus p_2) &= \mathrm{Cov}_u^{N_1}(p_1) \cdot \left(1 - \mathrm{Cov}_u^{N_1}(p_2)\right) \\ &\quad + (1 - \mathrm{Cov}_u^{N_1}(p_1)) \cdot \mathrm{Cov}_u^{N_1}(p_2) = 0.38 \end{aligned} \tag{5}$$

$$\begin{aligned}\mathrm{Pr}_{\mathrm{decode}}^{N_2}(p_1 \oplus p_2) &= \mathrm{Cov}_u^{N_2}(p_1) \cdot \left(1 - \mathrm{Cov}_u^{N_2}(p_2)\right) \\ &\quad + (1 - \mathrm{Cov}_u^{N_2}(p_1)) \cdot \mathrm{Cov}_u^{N_2}(p_2) = 0.5. \end{aligned} \tag{6}$$

Then, u's aggregate coding gain for the option $p_1 \oplus p_2$ can be obtained as follows:

$$\begin{aligned}\mathrm{Gain}_u(p_1 \oplus p_2) &= l(u,N_1) \cdot \mathrm{Pr}_{\mathrm{decode}}^{N_1}(p_1 \oplus p_2) + l(u,N_2) \cdot \\ &\quad \mathrm{Pr}_{\mathrm{decode}}^{N_2}(p_1 \oplus p_2) = 0.5 \times 0.38 + 0.8 \times 0.5 = 0.59. \end{aligned} \tag{7}$$

We can compute the aggregate coding gain of each possible coding option. The option with highest aggregate coding gain is chosen to be used for encoding and rebroadcasting.

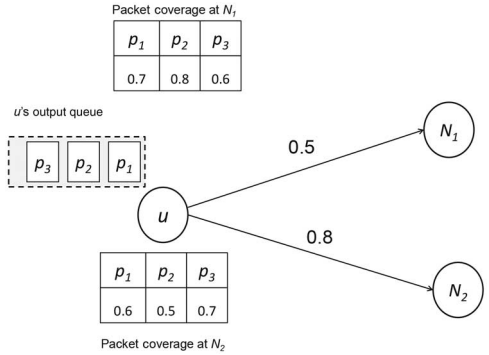


Fig. 5. Example of aggregate coding gain computation.

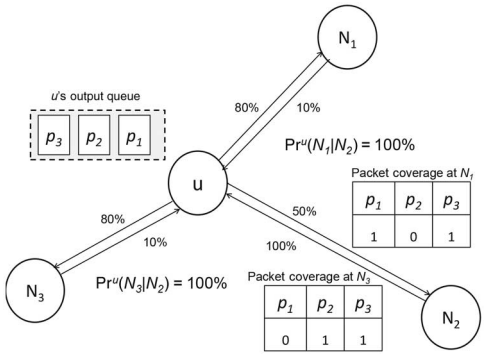


Fig. 6. Example of paged collective ACKs.

3) Optimization of Coding Option Set: When the page size is small, the size of coding option set is not very large. But when it becomes much larger, it might be too hard to choose the best coding option as the size of coding option set is too large. For the sake of optimizing the size of coding option set, we remove some near-zero-contribution coding options that statistically have near-zero contribution to the aggregate coding gain from the original coding option set by using the following two theorems.

Theorem 1: For a specific neighboring node k of the sender u ($k \in N(u)$) in a network, let $|\mathrm{pktsPool}(k)|$ denotes the number of native packets in k's packets pool that k has already decoded. If the maximum value of $|\mathrm{pktsPool}(k)|$ $(k = 1, \ldots, |N(u)|)$ is represented as $\mathrm{Max}|\mathrm{pktsPool}|$, a coding option Ω that encodes more than $\mathrm{Max}|\mathrm{pktsPool}| + 1$ packets cannot be decoded by any receiver and thus can be removed.

Theorem 2: Let $\bigcap \mathrm{pktsPool}(k)$ represents the the intersection set of $\mathrm{pktsPool}(k)$ $(k = 1, \ldots, |N(u)|)$ of all the neighboring nodes k. A coding option Ω whose coding set $\mathrm{natPktSet}(\Omega)$ is any subset of $\bigcap \mathrm{pktsPool}(k)$ will not have any contribution to the aggregated coding gain and thus can be removed as well.

B. Paged Collective ACKs

The mechanism of *paged collective ACKs* allows a node v to simultaneously infer the per-packet coverage probability $\mathrm{Cov}_v^k(p_i)$ of its neighbors $k \in N(v)$ for each native packet p_i in a specific page when receiving an encoded flooding packet.

The first component of *paged collective ACKs* is *page-based rebroadcasting mechanism*, a totally new approach taken by OppCode. In this mechanism, a node starts to rebroadcast only when it has received a whole page,while a node will compete to rebroadcast once receiving a single packet in existing single-packet flooding algorithms. The rationale behind this mechanism lies in that a node can make most effective coding decision only when it already has all packets in a page.

In OppCode, a transmission from sender u serves as two purposes from the view of v. 1) It is a direct ACK that u is a covered node, i.e., v can make sure that u has received the whole page, no matter v can decoded this transmission or not. 2) It is an implicit collective ACKs for v to update its neighbors' estimation of per-packet coverage $\mathrm{Cov}_v^k(p_i)$ by taking link correlation into consideration.

The mechanism of *paged collective ACKs* reduces redundant transmissions by considering link correlations. To clarify our idea, we consider a simple network including four nodes as shown in Fig. 6, where these nodes are within one-hop communication range of each other and u is assumed as covered while N_1, N_2, and N_3 are uncovered. If N_2 receives a transmission from u, N_2 can make sure that u is a covered node, while N_2 does not know the receiving status of N_1 and N_3 if link correlation has not been considered. However, if taking link correlation into consideration, this transmission also acts as collective ACKs to N_2 for N_1 and N_3, given that N_1 and N_3 have a reception probability $\Pr^u(N_1|N_2)$ and $\Pr^u(N_3|N_2)$, respectively. It should be noted that a transmission from u is an encoded packet as we employ network coding; thus, a collective ACK to N_2 can only implicitly ACK N_1 and N_3's receiving of this encoded packet but not native packets.

Based on the above network, we take a simple example to demonstrate the benefit of *paged collective ACKs* compared with *collective ACKs* in CF by decreasing the number of needed ACKs. Suppose u has packets p_1, p_2, and p_3 in its output queue and N_2's estimations of the packets that N_1 and N_3 have are p_1, p_3 and p_2, p_3, respectively, due to the diversity of packet loss among links. Without considering inter-packet dependance, CF needs u to keep broadcasting two packets p_1 and p_2 as implicit ACKs until N_2 makes sure $\mathrm{Cov}_{N2}^{N1}(2) = 1$ and $\mathrm{Cov}_{N2}^{N3}(1) = 1$ based on the assumption $\Pr^u(N_1|N_2) = 100\%$ and $\Pr^u(N_3|N_2) = 100\%$. Each packet needs to be transmitted twice due to $l(u, N_2) = 50\%$.

To address the problem described above, *paged collective ACKs* in OppCode allows a node to terminate transmission earlier if u chooses $p_1 \oplus p_2$ to be transmitted. Suppose that N_2 receives $p_1 \oplus p_2$ after two transmissions, and then it can immediately terminate its flooding mission due to $\mathrm{Cov}_{N2}^{N1}(2)$ and $\mathrm{Cov}_{N2}^{N3}(1)$, both get 1 based on the assumption $\Pr^u(N_1|N_2) = 100\%$ and $\Pr^u(N_3|N_2) = 100\%$. Therefore, the total number of transmissions can be reduced to 2, while the number is 4 in CF.

C. Tradeoff Between Link Correlation and Network Coding

From the above analysis, we have the idea that network coding can reduce the total number of transmission in broadcast when it has the opportunity to encode packets and link correlation can do the same thing while employing *collective ACKs*.

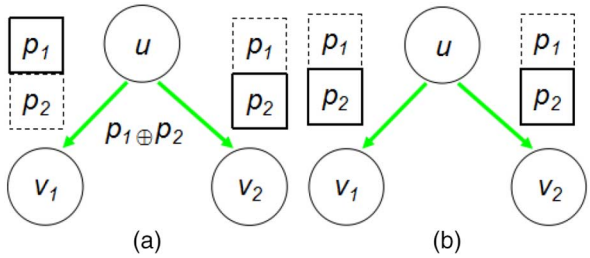


Fig. 7. Impact of link correlation on network coding. (a) Coding scenario. (b) Noncoding scenario.

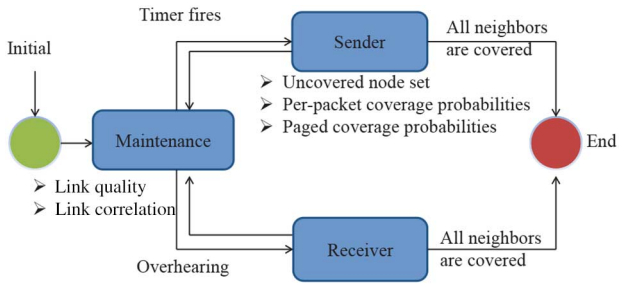


Fig. 8. FSM diagram of oppCode.

In OppCode, we combine network coding with link correlation, aiming at achieving the highest energy conservation. We found that there exists a tradeoff between link correlation and network coding. To illustrate this tradeoff, let us consider two extreme situations as follows. One is that a node u's neighbors are all negatively correlated (i.e., all the neighbors loss different packets), where network coding works best, as shown in Fig. 7(a), in which nodes v_1 and v_2 lose packet p_2 and p_1, respectively. The other is that u's neighbors are all positively correlated (i.e., all the neighbors loss the same packets), where network coding does not work, as shown in Fig. 7(b), in which nodes v_1 and v_2 all lose packet p_1. Comparing these two scenarios, we can find that there exist different coding opportunities with diverse link correlation conditions.

When the links in WSNs are highly correlated, aggregate-coding-gain-based mechanism is deserted and OppCode can flexibly change to noncoding protocol. Otherwise, we can use gain-based coding with correlated links. Link correlation has an impact on coding decision-making, and can guide us switching between coding and noncoding protocols. In OppCode, network coding is opportunistically decided based on link correlation. When the average link correlation reaches a threshold, we prefer not to code.

V. OppCode Flooding Protocol

This section describes the main design of OppCode protocol using a finite state machine (FSM), as shown in Fig. 8. After initializing stage, each node running OppCode protocol is in one of the following states: 1) maintenance; 2) sender; and 3) receiver state. Transitions between these states are triggered by events.

After initializing, a node first enters the maintenance state. Node in this state is responsible for updating link qualities and link correlations among neighbors. A node transits from maintenance state to sender state when its back-off timer fires. The node first computes aggregate coding gain of each coding option and then sends out an encoded packet with largest aggregate coding gain. After that, it updates the per-packet coverage probabilities of its neighbors and goes back to the maintenance state. Whenever the node receives a packet, it enters the receiver state and uses this packet as paged collective ACKs to update its neighbors' per-packet coverage probabilities in a batch, and then similarly returns to the maintenance state. When the node enters the maintenance state, it sets a back-off timer if the node has uncovered neighbors. Otherwise, this procedure terminates as the node estimates that all its neighbors have been covered.

A. Maintenance State

In maintenance state, every node periodically sends out a hello message identified by the node ID and a packet sequence number at an adaptive time interval T, which is decided by the condition of wireless environment. The hello messages are not only used for one-hop neighbor discovery but also for updating the link qualities and link correlations. The metric for link quality, denoted as $l(u,k)$, is defined as the ratio of the number of received packets from sender u by k, to the total number of packets sent by u, denoted as M.

The calculation of link correlation is more complex than that of link quality. Link correlation is defined as the probability that k receives a packet from sender u, given the condition that another node v receives this packet, denoted as $\Pr^u(k|v)$. For each receiver node, it keeps reception records of all hello messages from its neighbors, then we calculate link correlations as follows:

$$\Pr^u(k|v) = \frac{\sum_{i=1}^{M}(R_{uk}(i)\&R_{uv}(i))}{\sum_{i=1}^{M} R_{uv}(i)}. \tag{8}$$

Here, $R_{ux}(i)\,(x = v, k)$ is a bit representing x's reception status of the ith hello message sent from u. $R_{ux}(i) = 1$ if x receives this packet from u, otherwise, $R_{ux}(i) = 0$.

B. Sender State

A node enters the sender state when its back-off timer fires. A sender S should maintain three pieces of information.

1) Per-packet coverage probability $\mathrm{Cov}_u^k(i)$: $\mathrm{Cov}_u^k(i)\,(i = 1, \ldots, |\mathrm{Page}|;\ k \in N(u))$ indicates k's coverage probability by packet p_i from u's point of view.
2) Paged coverage probability $\overline{\mathrm{Cov}_u^k}$: $\overline{\mathrm{Cov}_u^k}$ is defined as the average coverage probability across a packet page, i.e.,

$$\overline{\mathrm{Cov}_u^k} = \frac{1}{|\mathrm{Page}|}\sum_{i=1}^{|\mathrm{Page}|}\mathrm{Cov}_u^k(i). \tag{9}$$

3) Uncovered node set $U(u)$: k is considered uncovered by u when $\overline{\mathrm{Cov}_u^k}$ is less than an application-specific threshold α. Here, $U(u) \subseteq N(u)$. Initially, u considers that all neighbors are uncovered, i.e., $U(u) = N(u)$.

After a node enters the sender state, it first computes the aggregate coding gain of each coding option according to (4) in Section IV, and chooses the one with highest value to be used for encoding and then send out the encoded packet. Then, it updates its neighbors' per-packet coverage probability $\mathrm{Cov}_u^k(i)$ of packet i grounded on

$$\mathrm{Cov}_u^k(i) \leftarrow \mathrm{Cov}_u^k(i) + l(u,k) \cdot \mathrm{Pr}_{\mathrm{decode}}^k(\Omega, i). \tag{10}$$

Equation (10) is composed of two terms. The first term denotes the probability that node k already has packet p_i before current transmission. The second one represents the contribution of current transmission, i.e., the probability that node k can decode the native packet p_i through current transmission from u, where $\mathrm{Pr}_{\mathrm{decode}}^k(\Omega, i)$ stands for the probability that node k can decode the packet p_i if this transmission of coding option Ω is received by k successfully.

Next, a more further step is to compute $\mathrm{Pr}_{\mathrm{decode}}^k(\Omega, i)$. The calculation of $\mathrm{Pr}_{\mathrm{decode}}^k(\Omega, i)$ depends on k's reception status from u's view, which has been described in Section IV-A2. For example, suppose u needs to flood a page that contains p_1, p_2, and p_3, it has its neighbor k's coverage probability $\mathrm{Cov}_u^k(i)$ of packet p_i $(i = 1, 2, 3)$ in store. If encoded packet $p_1 \oplus p_2$ is assumed as the best option, u will update u's per-page coverage probabilities after it sends out this encoded packet as follows:

$$\begin{aligned}
&\mathrm{Cov}_u^k(1) \leftarrow \mathrm{Cov}_u^k(1) + l(u,k) \cdot (1 - \mathrm{Cov}_u^k(1)) \cdot \mathrm{Cov}_u^k(2)\\
&\mathrm{Cov}_u^k(2) \leftarrow \mathrm{Cov}_u^k(2) + l(u,k) \cdot \mathrm{Cov}_u^k(1) \cdot (1 - \mathrm{Cov}_u^k(2))\\
&\mathrm{Cov}_u^k(3) \leftarrow \mathrm{Cov}_u^k(3).
\end{aligned} \tag{11}$$

When $\overline{\mathrm{Cov}_u^k}$ reaches a threshold α ($\alpha \leq 1$), k is considered as covered by u and removed from $U(u)$. If $U(u)$ gets empty, u terminates its flooding task. Otherwise, u returns to the maintenance state and then joins the forwarder competition again by setting its back-off timer. α is an application-specific parameter of OppCode. The higher the α is, the more reliable but less time- and energy-effective the OppCode will be.

C. Receiver State

A node enters receiver state once it receives or overhears a broadcasting encoded packet. For u, a rebroadcasting packet received from v serves as three purposes. First, since only node covered by a whole page can join the competition of rebroadcasting, u makes sure that v must have all the packets in this page and updates $\mathrm{Cov}_u^v(i)$ using $\mathrm{Cov}_u^v(i) = 1$.

Second, u tries to decode the received encoded packet and then updates its per-packet coverage probabilities $\mathrm{Cov}_u^u(i)$ based on the packets decoded. $\mathrm{Cov}_u^u(i)$ is a binary variable equal to 1 when u has decoded p_i successfully or otherwise equal to 0.

Finally, as the packet received also serves as paged collective ACKs, u can update its neighbors' coverage probabilities from its own point of view using the following equation:

$$\mathrm{Cov}_u^k(i) \leftarrow \mathrm{Cov}_u^k(i) + \mathrm{Pr}^v(k|u) \cdot \mathrm{Pr}_{\mathrm{decode}}^k(\Omega, i). \tag{12}$$

Here, $\mathrm{Pr}^v(k|u)$ is the probability that k can receive a packet on the condition that node u has received the same packet. The term of $\mathrm{Pr}^v(k|u) \cdot \mathrm{Pr}_{\mathrm{decode}}^k(\Omega, i)$ shows the probability that its neighbor k can receive and decode the native packet p_i from u's point of view.

As in sender state, when $\overline{\mathrm{Cov}_u^k}$ reaches α, k is considered by u as covered and removed from uncovered node set $U(u)$ of u. If $U(u)$ is not empty and u is covered by all packets in the specific page, u will join the competition for being next local forwarder by setting back-off timer. Otherwise, u exits receiver state and completes its rebroadcasting mission.

D. Back-Off Timer Design

The back-off timer is used to conduct dynamic forwarder competition, whose duration is set according to many factors such as uncovered neighborhood size and link qualities. In OppCode, *maximum gain* (or MG) is defined as a metric for setting the back-off timer.

Definition 1: Maximum gain: MG*(u)* equals the maximal aggregate coding gain among all possible coding options for a sender u, i.e., the maximal expected number of all decoded native packets across the neighborhood if u transmits once.

Specifically, the value of MG*(u)* can be calculated using the following equation:

$$\mathrm{MG}(u) = \mathrm{MAX}_{\Omega}(\mathrm{Gain}_u(\Omega)). \tag{13}$$

Intuitively, the higher the $\mathrm{MG}(u)$ value is, the more effective the u's forwarding is and the smaller its back-off timer period should be. The rationale behind this is that we always select the forwarder which is able to make more nodes in the network able to decode native packets with one transmission.

Each node u updates its $\mathrm{MG}(u)$ dynamically during the dissemination of the packets as it updates the value of per-packet $\mathrm{Cov}_u^k(i)$ whenever it sends or receives an encoded packet from its neighbors.

E. Detailed Protocol

Combining all the design components, OppCode can be described by the pseudocode shown in Algorithm 1. The algorithm is simple and fully distributed that requires only one-hop local information.

In OppCode, each node has an FSM with three states, i.e., maintenance, sender, and receiver states. State transitions are triggered by the events of either sending timer firing (line 4) or receiving a broadcast packet (line 11). Lines 5 to 10 handle the event of sending timer firing. Lines 12 to 19 handle the event of receiving a packet. Lines 20 to 22 update the uncovered set of a node, and lines 23 to 26 determine whether the flooding task should be terminated or not.

To sum up, the OppCode protocol has three key features. 1) It can be implemented with a simple three-state FSM, which is resource-efficient and thus suitable for resource constrained sensor nodes. 2) It deals with dynamic forwarder competition with a metric of maximum gain $\mathrm{MG}(u)$ for each sender u. 3) It reduces the communication redundancy through paged collective ACKs, eliminating costly per-packet ACKs from every receiver.

Algorithm 1. OppCode

```
Initially, U(u) ← N(u); ∀ k ∈ U(u) and ∀ i from 1 to |Page|, Cov_u^k(i) ← 0;
While U(u) ≠ ∅ do
    Switch Event do
        case timer fired
            u encodes and sends using option with highest Gain_u(Ω) via Equation. 1;
            for k ∈ U(u) do
                for i from 1 to |Page| do
                    Update Cov_u^k(i) via Equation. 2;
                Call Update U(u);
            Call Test U(u);
        case u receives packet from v
            for k ∈ U(u) do
                if k==v then for i from 1 to |Page| do
                    Cov_u^k(i) = 1;
                else
                    for i from 1 to |Page| do
                        Updates Cov_u^k(i) via Equation 3;
                Call Update U(u);
            Call Test U(u);
Update U(u) function:
if Cov_u^k ≥ α then
    U(u) ← U(u) − k
Test U(u) function:
if U(u) ≠ ∅ then Set back-off timer via Equation 4;
else
    Terminate the timer;
```

VI. Simulation

In order to better understand the performance of OppCode, we compare its performance with the following three solutions with extensive simulation results.

1) CF by Zhu *et al.* in NSDI'10.
2) OppCode with random coding decision (OppCode-R): OppCode-R works similarly with OppCode except making coding decision randomly from all possible coding options.
3) Oracle: Oracle works similarly with OppCode, whose only difference is that every node *exactly* knows which part of packets its neighbors have. One can imagine that there are cost-free ACKs that can be employed by nodes to exchange its received packet list with their neighbors.

Four metrics are used to evaluate the protocols.

1) Reliability: Reliability is measured by the percentage of nodes that received the whole flooding page in a network.
2) Transmission overhead: Transmission overhead is quantified by the total number of transmissions by all nodes, excluding hello messages in initialization period.
3 Flooding delay: Flooding delay is the time period from the time that the source initiates the flooding to the time when no more nodes rebroadcast.
4) Load balance: Load balance is a metric measuring the uniformity of the rebroadcasting activities distribution across the network, which is defined by the standard deviation of the number of transmissions per node.

A. Simulation Setup

Our design is implemented based on the network simulation tool OMNET-4.1. In our simulation experiments, a standard CSMA/CA protocol without ACKs and retransmissions is adopted as the MAC layer. The radio model is implemented based on our empirical data, which has obvious link-correlation feature as described in Section III-A.

We use randomly generated network topologies to evaluate our design. In the simulation, we randomly deploy 250 sensor nodes in a 1000 m $\times$1000 m square field and the communication range is set within 155 m. A source node is positioned at one corner of the field, which sends out a packet with 29-byte payload every 5 s. The total simulation time is set to 1050 s. The first 50 s is network initialization period, in which nodes only exchanges hello messages between neighbors to establish the neighborhood information. The source keeps sending out 200 data packets from 50 to 1050 s. Every data point stands for the averaged value of results over 10 runs. Unless explicitly declared, the above default values are used in our simulation experiments.

B. Impact of Node Density

We first analyzes the effect of node density on the protocol performances by varying the number of nodes from 100 to 250 nodes. Fig. 9(a) shows that OppCode performs much better compared with CF and OppCode-R and its performance is very close to that of Oracle, although these protocols all have high reliability more than 95% as the node density increases. The reliability performances of all these protocols increase as the network density increases. More nodes the network has, more opportunities a node has to be connected with neighbors and covered by flooding packets, and thus higher reliability it can obtain. Besides, the mechanism of dynamic forwarder selection contributes to achieve high reliability. While in traditional fixed-forwarder approaches, if these forwarders fail, the uncovered nodes cannot be covered anymore.

Fig. 9(b) shows that the total number of transmissions of all the protocols increases as the network gets denser. It can be seen that total number of transmissions only increases slightly in OppCode. The reason is that as the node density increases, more neighbors help a node to be covered by predicting its per-packet coverage probability, which results in a decrease in per-node number of transmissions a node needs and only a slight increase in total number of transmissions. It is not hard to find that OppCode has much less total number of transmissions compared with CF whatever the node density is. This is because in OppCode, the technique of network coding is employed, each time the node dynamically selects the best coding option for the current transmission; thus, OppCode has much more efficient rebroadcasts and less transmissions.

Fig. 9(c) shows that when the network density increases, the end-to-end delay decreases in all these four protocols. The reason is simple that the number of per-node retransmissions

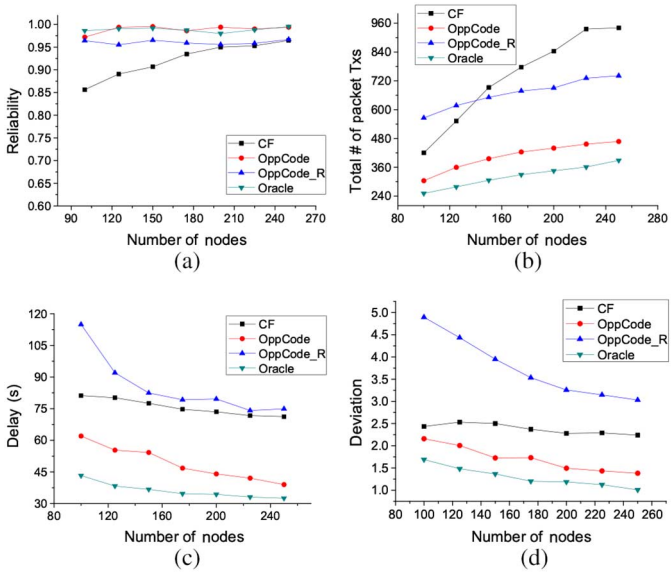


Fig. 9. Impact of node density. (a) Reliability. (b) Transmission overhead. (c) Flooding delay. (d) Load balance.

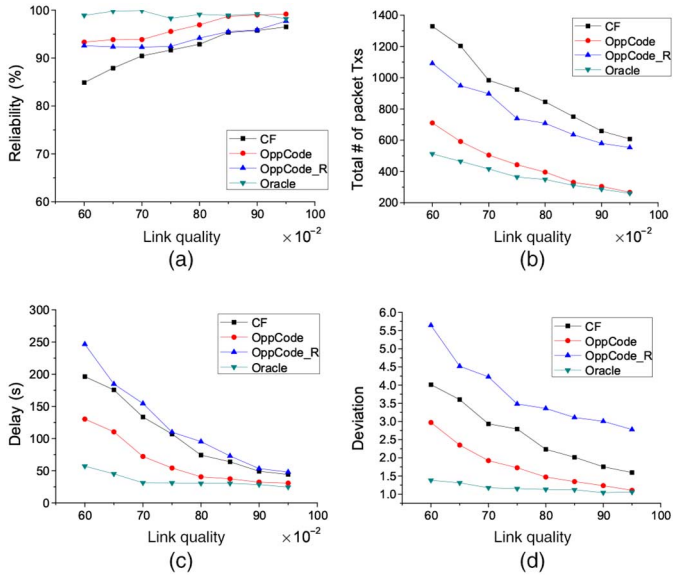


Fig. 10. Impact of unreliable links. (a) Reliability. (b) Transmission overhead. (c) Flooding delay. (d) Load balance.

used in these protocols decreases when the network density increases. OppCode-R performs worst here due to the inefficiency of the random coding decision it employs.

Fig. 9(d) shows that when the network density increases, the standard deviations of the number of transmissions per node for all protocols decrease, because that denser the network is, less nonuniformity of transmission number exists among nodes. OppCode also performs better compared with CF and OppCode-R, which achieves more even distribution of transmissions across the network.

C. Impact of Unreliable links

In this experiment, we analyze the effects of varying link qualities on protocol performances. The average link quality varies from 60% to 100% for each experiment trial.

Fig. 10(a) shows that as the link quality increases, the reliability increases for all the protocols. The reliability of OppCode and OppCode-R is higher than 90% when the link quality varies from 0.6 to 1, while CF has lower reliability less than 90% when the link quality is low. As a result, OppCode and OppCode-R are better suitable for high-reliability applications in WSNs with low link quality. Additionally, as shown in Fig. 10(b), OppCode costs lower energy for data transmission, since OppCode employs network coding to further reduce redundant transmission compared with CF. Compared with OppCode-R that randomly chooses a coding option, OppCode selects the best option for next transmission each time, which results in the decrease in the number of transmissions.

Fig. 10(c) shows that the end-to-end delay decreases for all protocols as the link quality increases. This is because better the link quality is, fewer retransmissions needed by all the protocols. Fig. 10(d) shows that as the link quality increases, the standard deviation of the number of transmissions per node for all protocols decreases, because that better the link quality is, less nonuniformity of transmission number exists among nodes.

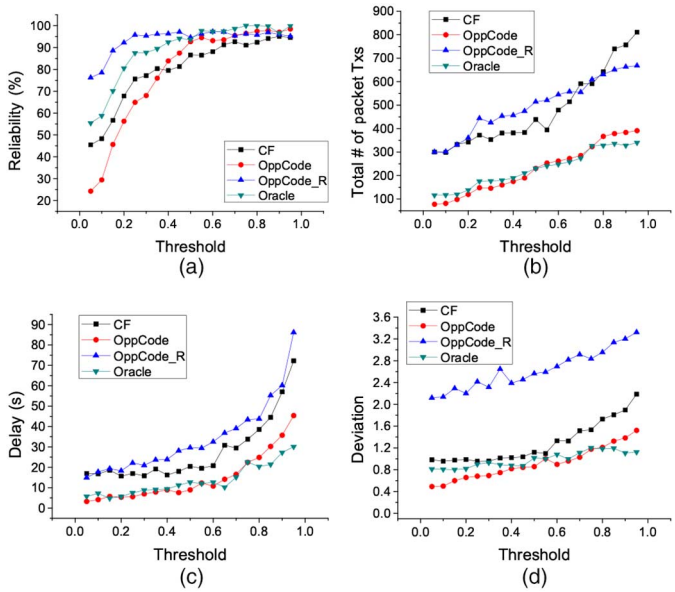


Fig. 11. Impact of reliability requirements. (a) Reliability. (b) Transmission overhead. (c) Flooding delay. (d) Load balance.

D. Impact of Reliability Requirements

CF, OppCode, and OppCode-R all use a coverage threshold α to control the reliability users need. We then analyze the impact of α with the setup that the total number of nodes is 150, and the α value varies from 0.1 to 0.9 with step 0.1.

Fig. 11(a) shows that the reliability of all the protocols increases when the α value increases. We note that the reliability of all the protocols is almost always above the line with the angle of 45, indicating that these protocols all satisfy the users' requirement well.

Fig. 11(b) and (c) shows when the α value increases, the total number of transmissions increases for all the protocols as well as transmission delay. OppCode helps to reduce the number of transmissions while generating lower delay, compared with CF

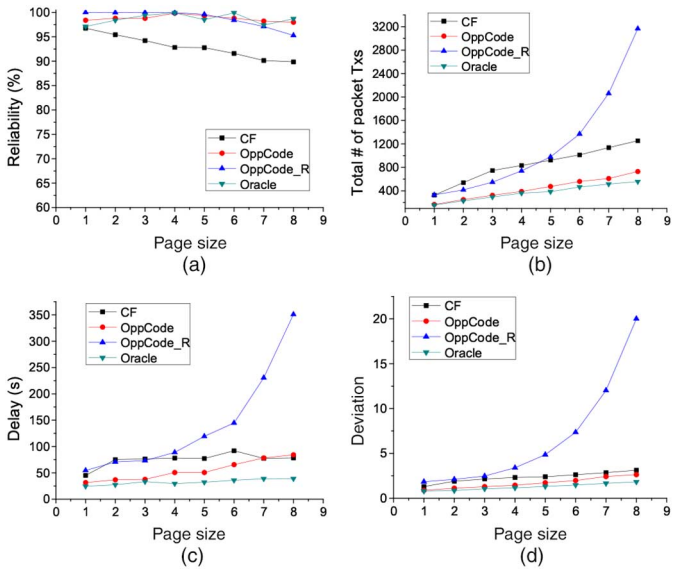


Fig. 12. Impact of page size. (a) Reliability. (b) Transmission overhead. (c) Flooding delay. (d) Load balance.

and OppCode-R. Fig. 11(d) shows as the α value increases, the standard deviation of the number of transmissions per node for all the protocols increases to some extent.

E. Impact of Page Size

Finally, we analyze the impact of page size on the performance of four flooding protocols. The page size varies from 1 to 8 for OppCode, OppCode-R, and Oracle, while the number of independent flooding packets varies from 1 to 8 for CF.

Fig. 12(a) shows that when the page size increases, OppCode and OppCode-R achieve high reliability above 95%, while the reliability of CF decreases to below 90% in some points. The result indicates that CF is not suitable to be applied to high-reliability multipacket flooding applications, while OppCode is best suitable for this kind of flooding applications.

As shown in Fig. 12(b), the total number of transmissions slightly increases in CF and OppCode when the page size increases, while the number increases dramatically in OppCode-R. The reason is that the increasing page size leads to a huge size of the coding option set, and thus random network coding causes huge amount of ineffective transmissions. OppCode uses greedy coding decision instead of random scheme, which is much more effective and helps to reduce the number of transmission. Besides, OppCode combines network coding with link correlation, which contributes to the decrease in the number of transmissions compared with CF.

Fig. 12(c) and (d) shows that both OppCode and CF have lower end-to-end delay and better load balance compared with OppCode-R. With comprehensive analysis about Fig. 12(a)–(d), we conclude that OppCode is the best achieving energy- and time-efficiency compared with CF and OppCode-R while having higher reliability and better load balance.

VII. Implementation and Evaluation

To evaluate the real-world performance of our design, we compare the performance of OppCode with CF [10] and

Fig. 13. Indoor WSNs testbed.

standard flooding (FLD) protocol, in which each node rebroadcasts its first-time received packet exactly once, using four performance metrics: reliability, message overhead, flooding delay, and load balance. All these protocols are implemented based on TinyOS 2.1.0. AMSenderC and AMReceiverC are the main communication-related TinyOS components used and FTSP is adopted as the time-synchronization protocol.

A. Experimental Setup

The experiments are conducted on an indoor 2.5 m $\times$ 6.0 m testbed consisted of 21 TelosB nodes, as shown in Fig. 13. In the experiments, the transmission power of each node is tuned down to level 2 to ensure that multihop network topology can be formed. Without loss of generality, the page size for each node is fixed as four. To reduce the influence of randomness in experiments, each result is obtained averaged over 15 runs.

In a 50-s network initialization phase, all nodes get synchronized and then start the neighbor discovery by exchanging hello messages. After that, all the nodes have obtained the link quality and link correlation information about their one-hop neighbors. Then, a node is selected as the sender to send out 100 data packets with a time interval of 10 s. For performance analysis purposes, in each data packet, we include information such as hop count, time stamp, and the previous hops node ID. Unless explicitly declared, the above default values are used in all the experiments.

B. Experimental Performance

We analyze the average performance of these three flooding protocols and the impact of varying reliability threshold α on their performance of protocols as shown in Figs. 14 and 15.

Fig. 14(a) shows that the average reliability of FLD, CF, and OppCode is 53.5%, 99.65%, and 99.6%, respectively. OppCode and CF reach high reliability more than 99%, much higher than FLD. Furthermore, OppCode has fewer transmissions and shorter delay than CF does as shown in Fig. 14(b) and (c). The average values of the total number of transmissions and flooding delay in seconds for FLD, CF, and OppCode are 42.8, 56.88, 50.32 and 1.82, 1.63, 1.5, respectively. Thus, OppCode almost reduces the total number of transmissions and flooding delay by 11.53% and 7.98%, respectively, while achieving the same reliability compared with CF. These improvements are due to opportunistic coding that allows for a more efficient use of the wireless media. Especially, OppCode works best when the links are perfectly negative correlated, i.e., different links

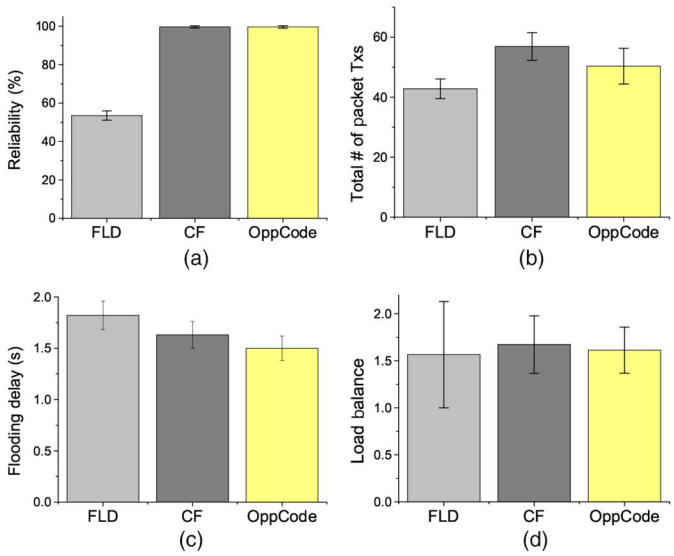


Fig. 14. Performance of protocols in indoor experiments. (a) Reliability. (b) Transmission overhead. (c) Flooding delay. (d) Load balance.

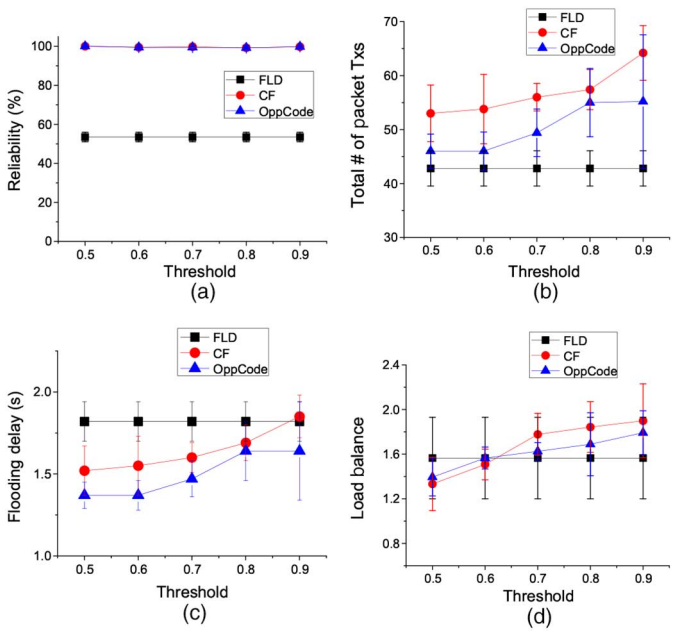


Fig. 15. Impact of reliability threshold in indoor experiments. (a) Reliability. (b) Transmission overhead. (c) Flooding delay. (d) Load balance.

lost different packets, reducing the number of transmissions by sending encoded packets. In other words, if the links are perfectly positive correlated, i.e., different links lost the same packets, the total number of transmissions would be the same with or without network coding.

Fig. 15(a)–(c) shows the reliability, message overhead, and flooding delay of FLD, CF, and OppCode as the reliability threshold α various from 0.5 to 0.9. In Fig. 15(b) and (c), both the total number of transmissions and flooding delay in CF and OppCode increase as the reliability threshold α increases, while OppCode always has fewer transmissions and shorter delay compared with CF, which shows that OppCode is more suitable for multipacket flooding in energy-sensitive or time-sensitive applications.

Figs. 14(d) and 15(d) show the performance of load balance of FLD, CF, and OppCode. While the threshold α varies from 0.5 to 0.9, OppCode has a slightly higher value of standard deviation than CF before threshold 0.63, while it has lower value after threshold 0.63. Thus, OppCode is more suitable when applied to flooding applications requiring high reliability.

C. Island-Node Observation

While conducting real-world experiment, we observe an *island-node* phenomena and propose a practical method to reduce unnecessary transmissions and additional energy consumption induced by them.

Island-node observation: In flooding, a node with very low link qualities in a network keeps generating useless rebroadcasts, which contributes very little to its neighbors' coverage.

We consider a node exhibiting this observation as an *island-node*. An *island-node* keeps generating large amount of unnecessary disseminations with little contribution to the coverage of its neighbors. This is because of two reasons.

1) Each transmission of an *island-node* has a very low increase of its estimation of its neighbors' coverage probability due to its low link qualities to the neighbors.
2) An *island-node* may not receive its neighbors' rebroadcasts as paged collective ACKs to update its estimation of its neighbors' coverage probability.

In order to make its neighbors covered in its own point of view, it will keep retransmitting packets until the paged coverage of each neighbor gets higher than the threshold α.

In the implementation of OppCode, to address the *island-node* problem, we set a limit on the number of retransmissions for each node avoiding that few *island-node*s lead to high number of rebroadcasts and long end-to-end flooding delay of the whole network. The experimental results show that this solution with a retransmission limit works well, which significantly reduces the number of useless transmission by *island-node*s, while high reliability still can be achieved.

VIII. Conclusion

In this paper, we present the design of a network-coding-based multipacket flooding protocol OppCode that provides efficient and reliable message dissemination service for WSNs with unreliable and correlated links. We demonstrate that OppCode is effective through two key mechanisms: *opportunistic coding decision* and *paged collective ACKs*. We evaluated its performance based on both simulations and real-world experiments compared with that of state-of-the-art protocols. Experimental results show that OppCode has high reliability, low transmission overhead, low delay, and good load balance.

References

[1] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. ACM SenSys'04*, 2004, pp. 81–94.
[2] G. S. M. Maroti, B. Kusy, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. ACM SenSys'04*, 2004, pp. 39–49.
[3] F. Gandino, B. Montrucchio, and M. Rebaudengo, "Key management for static wireless sensor networks with node adding," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1133–1143, May 2014.

[4] O. Gnawali *et al.*, "Collection tree protocol," in *Proc. ACM SenSys'09*, 2009, pp. 1–14.

[5] J. Niu, L. Cheng, Y. Gu, L. Shu, and S. Das, "R3E: Reliable reactive routing enhancement for wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 784–794, Feb. 2014.

[6] D. Zhang, G. Li, K. Zheng, X. Ming, and Z. Pan, "An energy-balanced routing method based on forward-aware factor for wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 766–773, Feb. 2014.

[7] W. Lou and J. Wu, "Double-covered broadcast (DCB): A simple reliable broadcast algorithm in manets," in *Proc. IEEE INFOCOM'04*, 2004, pp. 2084–2095.

[8] F. Stann *et al.*, "RBP: Robust broadcast propagation in wireless networks," in *Proc. ACM SenSys'06*, 2006, pp. 85–98.

[9] K. Srinivasan *et al.*, "The k factor: inferring protocol performance using inter-link reception correlation," in *Proc. ACM MobiCom'10*, 2010, pp. 317–328.

[10] T. Zhu *et al.*, "Exploring link correlation for efficient flooding in wireless sensor networks," in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation (NSDI'10)*, 2010, pp. 49–64.

[11] S. Wang *et al.*, "CorLayer: A transparent link correlation layer for energy efficient broadcast," in *Proc. ACM MobiCom'13*, 2013, pp. 27–38.

[12] X. Cao, J. Chen, Y. Zhang, and Y. Sun, "Development of an integrated wireless sensor network micro-environment monitoring system," *ISA Trans.*, vol. 47, no. 3, pp. 247–255, 2008.

[13] J. Wang, C. Lin, E. Siahaan, B. Chen, and H. Chuang, "Mixed sound event verification on wireless sensor network for home automation," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 803–812, Feb. 2014.

[14] D. Zhang, "A new approach and system for attentive mobile learning based on seamless migration," *Appl. Intell.*, vol. 36, no. 1, pp. 75–89, 2012.

[15] D. Zhang and X. Zhang, "Design and implementation of embedded un-interruptible power supply system (EUPSS) for web-based mobile application," *Enterp. Inf. Syst.*, vol. 6, no. 4, pp. 473–489, 2012.

[16] D. Zhang and Y. Liang, "A kind of novel method of service-aware computing for uncertain mobile applications," *Math. Comput. Model.*, vol. 57, no. 3, pp. 344–356, 2013.

[17] I. H. Hou *et al.*, "AdapCode: Adaptive network coding for code updates in wireless sensor networks," in *Proc. 27th IEEE Conf. Comput. Commun. (INFOCOM'08)*, 2008, pp. 2189–2197.

[18] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes," in *Proc. Int. Conf. Inf. Process. Sensor Netw. (IPSN'08)*, 2008, pp. 457–466.

[19] S. Kulkarni and L. Wang, "MNP: Multihop network reprogramming service for sensor networks," in *Proc. 25th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS'05)*, 2005, pp. 7–16.

[20] M. Rossi *et al.*, "SYNAPSE++: Code dissemination in wireless sensor networks using fountain codes," *IEEE Trans. Mobile Comput.*, vol. 9, no. 12, pp. 1749–1765, Dec. 2010.

[21] S. He, J. Chen, D. K. Yau, and Y. Sun, "Cross-layer optimization of correlated data gathering in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 11, pp. 1678–1691, Nov. 2012.

[22] D. Zhang and Y. Zhu, "A new constructing approach for a weighted topology of wireless sensor networks based on local-world theory for the Internet of Things (IOT)," *Comput. Math. Appl.*, vol. 64, no. 5, pp. 1044–1055, 2012.

[23] D. Zhang and C. Zhao, "A new medium access control protocol based on perceived data reliability and spatial correlation in wireless sensor network," *Comput. Elect. Eng.*, vol. 38, no. 3, pp. 694–702, 2012.

[24] J. Luo, J. Hu, D. Wu, and R. Li, "Opportunistic routing algorithm for relay node selection in wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp. 112–121, Feb. 2015.

[25] R. Ahlswede *et al.*, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[26] S. Katti *et al.*, "XORs in the Air: Practical wireless network coding," in *Proc. ACM SIGCOMM'06*, 2006, pp. 243–254.

[27] J. Subramanian *et al.*, "UFlood: High-throughput flooding over wireless mesh networks," in *Proc. IEEE INFOCOM*, 2012, pp. 82–90.

[28] Y. Zhang *et al.*, "Opportunistic coding for multi-packet flooding in wireless sensor networks with correlated links," in *Proc. IEEE 11th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS'14)*, 2014, pp. 371–379.

[29] S. I. Alam, S. Sultana, Y. C. Hu, and S. Fahmy, "SYREN: Synergistic link correlation-aware and network coding-based dissemination in wireless sensor networks," in *Proc. IEEE 21st Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst. (MASCOTS'13)*, 2013, pp. 485–494.

[30] S. Wang *et al.*, "Efficient network coding under correlated unreliable wireless links," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP'14)*, 2014, pp. 1–12.

Xingfa Shen (M'10) received the B.S. degree in electrical engineering, and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2000 and 2007, respectively.

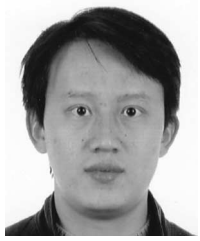
Currently, he is working as an Associate Professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou. His research interests include wireless sensor networks, Cyber-Physical Systems (CPS), and mobile computing.

Yueshen Chen received the B.S. degree in computer science from Hangzhou Dianzi University, Hangzhou, China, in 2013. Currently, he is pursuing the Master's degree at the School of Computer Science and Technology, Hangzhou Dianzi University.

His research interests include wireless sensor networks and mobile computing.

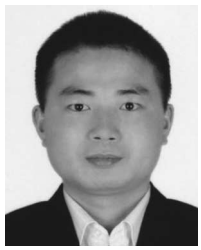
Yinqun Zhang received the B.S. degree in computer science from Dalian Nationalities University, Dalian, China, in 2012. Currently, she is pursuing the Master's degree at the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China.

Her research interests include wireless sensor networks and Cyber-Physical Systems (CPS).



Jianhui Zhang (M'09) received the B.S. degree in mechanotronics and the M.S. degree in fluid mechanics from Northwestern Ploytechnical University, Xi'an, China, in 2000 and 2003, respectively, and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2008.

Currently, he is working as an Associate Professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou. His research interests include algorithm design and wireless networks.



Quanbo Ge (M'08) received the B.S. and M.S. degrees in computer science from Henan University, Kaifeng, China, and the Ph.D. degree in electrical engineering from Shanghai Maritime University, Shanghai, China, in 2002, 2005, and 2008, respectively.

Currently, he is working as an Associate Professor with the School of Automation, Hangzhou Dianzi University, Hangzhou, China. His research interests include data fusion and wireless sensor networks.

Guojun Dai (M'00) received the Ph.D. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 1998.

Currently, he is working as a Professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou. His research interests include embedded systems and electrical control.

Tian He (M'02) received the Ph.D. degree in computer science from the University of Virginia, Charlottesville, VA, USA, in 2004.

Currently, he is working as an Associate Professor with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA. His research interests include wireless sensor networks, cyber-physical systems, and intelligent transportation systems.