International Journal of Computer Engineering Research Vol. 2(2), pp. 34-44, March 2011
Available online at http://www.academicjournals.org/IJCER
ISSN 2141- 6494 ©2011 Academic Journals

Full Length Research Paper

Efficient power savings in wireless sensor networks with network coding and overhearing avoidance

Hnin Yu Shwe* and Xiaohong Jiang

Electrical and Communication Engineering, Graduate School of Engineering, Tohoku University, 6-6-05 Aza-Aoba, Aramaki, Aoba-ku, Sendai, 980-8579 Japan.

Accepted 24 February, 2011

Wireless sensor networks and embedded systems are becoming commonplace in many fields of research. However, sensor nodes are generally powered by batteries which provide a limited amount of power, and it is often difficult to recharge or replace the batteries. Therefore, power aware and power efficient protocols at each layer of the communications are very important for wireless sensor networks. Various schemes tried to achieve significant power saving, among them; the network coding is one promising technique. To the best of our knowledge, AdapCode is by now, the most promising network coding architecture for power saving in wireless sensor networks. However, the AdapCode has two clear limitations. First, the coding process of AdapCode works based only on partial neighbors of a node which may waste some potential coding opportunities. Second, the AdapCode did not consider overhearing and idle listening issues in packets forwarding, so, it may cause unnecessary battery power expenditure from receiving irrelevant packets. To address the earlier mentioned limitations, we first propose an improved AdapCode scheme (namely AdapCode+) by deploying a power efficient protocol to discover all neighbors of a node, such that the potential coding opportunities can be increased. To further avoid overhearing irrelevant packets, we then enhance AdapCode+ to AdapCode++. The basic idea of AdapCode++ is to compose new digest information about the subsequent packets to be sent out from current node, and then broadcast it together with the conventional wake up message to all the neighbor nodes. In this way, the neighbor nodes can intelligently determine receiving desired packets only. Our NS-2-based simulation indicates that the simple AdapCode+ scheme can improve the AdapCode in terms of power saving, and the AdapCode++ can further make this improvement much more significant.

Key words: Sleep-wake scheduling, duty-cycling, overhearing, power saving, network coding, wireless sensor network.

INTRODUCTION

Explosive growth in embedded computing and rapid advances in low power wireless networking technologies are fueling the development of wireless sensor networks (WSNs). WSN may consist of even thousands of small and fully autonomous nodes, which gather sensor information, perform data processing, and communicate with each other. The development of the WSNs was originally motivated by the military service application areas like battlefield surveillance, however, they are now also widely used in many application areas such as environmental monitoring, habitat monitoring and mission critical networks.

In WSNs, each sensor node has the capability of sensing particular physical phenomena in its vicinity and communicating with other nodes in the same network using wireless transceivers. Broadcasting is a commonly used mechanism for disseminating identical information from one source to many receivers. However, as sensor nodes are usually battery-powered, one of the major limitations on performance and lifetime of such networks is the limited capacity of these finite power sources.

*Corresponding author. E-mail: hninyu@mobile.ecei.tohoku.ac.jp.

Since the most power consumption action for WSN is the data communication, in order to make optimal use of power consumption, communication should be minimized as much as possible.

Numerous researchers have recently suggested a variety of mechanisms for achieving power-efficiency in sensor networks, namely, in-network processing, data aggregation, and network coding. One of the useful approaches is in-network processing or data aggregation (Demirkol et al., 2006; Li and Li, 2004). In such a setting, certain nodes in the sensor network, called aggregators, collect the raw information from the sensors, process it locally, and reply to the aggregate queries of the central server (Son et al., 2007). However, this approach cannot be used when all the original packets are needed at the received nodes. Recently, network coding became a new approach that shows an awareness of the power consumption by increasing the transmission capacity of a network (Akyildiz et al., 2002; Fragouli et al., 2006).

In a traditional store-and-forward network, packets are forwarded hop-by-hop along the intermediate nodes (for example, routers) from a source to a destination. An intermediate node forwards the packets as it receives through a predefined path. On the other hand, network coding technique allows an intermediate node to combine data from different input links before sending the combined data to its output links. Hence, for many problems such as multicast and broadcast, using appropriate encoding schemes at each intermediate node can achieve the network capacity. Moreover, some research works have already shown that network coding technique can also be applied to wireless networks, thus, in recent days, researchers have been investigating on how sensor networks can get benefits from network coding.

To the best of our knowledge, the AdapCode (Haenggi, 2004) is by now the most promising network coding architecture for power saving in WSNs. The main idea of AdapCode is to adaptively adjust the coding combination of packets in a node according to the number of its neighbor nodes, so, AdapCode's performance highly depends on how many neighbors a node has. Generally, if a node has more neighbors, it can encode more packets together without losing reliability since it can easily get enough combinations from its neighbors to decode the original packets.

Although the available AdapCode scheme is good at power saving from using network coding, it has two clear problems that may significantly limit its efficiency. First, it considers only partial neighbor nodes (namely full-active neighbor nodes) while determining the coding combinations, which may waste some potential coding opportunities. Second, it suffers from the problem of overhearing and idle listening. This is due to the broadcast nature of the wireless channel where all nodes in the vicinity of a sender node overhear its packet transmissions even if they are not the intended recipients of these transmissions. This kind of redundant reception will result in an unnecessary power expenditure of the recipients.

Based on these observations, in this paper, we first enhance the AdapCode to AdapCode+ by deploying a power efficient neighbor discovery protocol to find out all the neighbors of a node, such that its potential coding opportunities can be increased. To address the second problem of AdapCode, we further improve the AdapCode+ to AdapCode++, in which the redundant overhearing of irrelevant packets is eliminated through using new digest information together with the wakeup message. We will show that through such improvements, the power saving performance of AdapCode can be significantly enhanced.

OVERVIEW OF AdapCode

AdapCode architecture

It is notable that among the network coding architectures for WSN proposed by now, the AdapCode scheme is the most promising one because, each node can adaptively choose the coding combination of packets depending on the number of its neighbor nodes. The AdapCode is a reliable data dissemination protocol, using adaptive network coding, and it can reduce traffic in the process of code update.

Taking advantage of variations in connectivity, the AdapCode presents an adaptive network coding protocol, where nodes dynamically decide the new coding choice, N, based on how many neighbors they have. The coding methodology in AdapCode is to randomly generate N coefficients and compute the linear combination of N packets. Gaussian elimination is used to decode the original packets.

Parameter selection

There are two parameters to be considered in network coding (Polastre et al., 2004). The first one is how many data segments, N, are proper to form a group. If N is too small, it is unable to take the advantage of potential network coding capability. On the other hand, if N is too large, a node might not receive the required number of packets to decode the original packet. The second parameter is how many packets are needed to deliver for obtaining the desired reliability. Different applications may have different optimal value.

Table 1 shows the choice of N used in the AdapCode according to the average neighbors of a node. For each M received packets, each node will combine N packets together into one coded packet and then will broadcast the M/N coded packets only.

Table 1. The choice of N according to *avg Neighbor.*

avgNeighbor	0-4	5-7	8-10	11~
N	1	2	4	8

Limitations of the available network coding algorithm in AdapCode

The current network coding architecture of AdapCode is quite simple to maintain. However, it has the following limitations: 1) In AdapCode, due to the broadcast nature of communication, each node can opportunistically overhear the packets which have already been received before. Receiving those redundant packets will result in unnecessary power dissipation and this is especially significant for dense sensor networks, where each node has several neighbors in its close vicinity; 2) we should notice that if a node has more neighbors, it can get more coding opportunities since it has the potential to get enough messages from its neighbor nodes. However, the current structure cannot fully explore this potential, because the AdapCode did not count some nodes as the neighbors of it. More specifically, if nodes have more neighbors, they can encode more packets together since they get more than enough combinations from their neighbors to reconstruct the original message. Therefore, the AdapCode will significantly limit the potential coding opportunities.

To illustrate the limitation of current neighbor discovery structure, we consider a simple example as shown in Figure 1. Here, for the sake of simplicity, we define the full-active and semi-active neighbor nodes used in our schemes. We will say that node B is a full-active neighbor of node A if node B is in the transmission range of node A and node B has already sent some messages to node A in previous time stamp. On the other hand, if node B is in the transmission range of node A but it did not send any message to node A earlier, we will regard node B as a semi-active neighbor of node A.

In this example, although the actual number of neighbors for sensor S is 6, the AdapCode will consider the number of neighbors only just as 4 without considering the semi-active neighbors because it received messages from only those 4 full-active nodes in previous time-stamp, but not from the other 2 semi-active nodes. Finally, the sensor S will choose its new coding choice, N, as 1 according to the specified table. It means, in such case, the sensor S will not perform network coding and it will broadcast the naive packet as it is, and resulting in the loss of coding opportunity.

To address the above limitations of the available network coding structure in AdapCode, we propose a new neighbor discovery algorithm and overhearing avoidance network coding algorithm in further discussions.

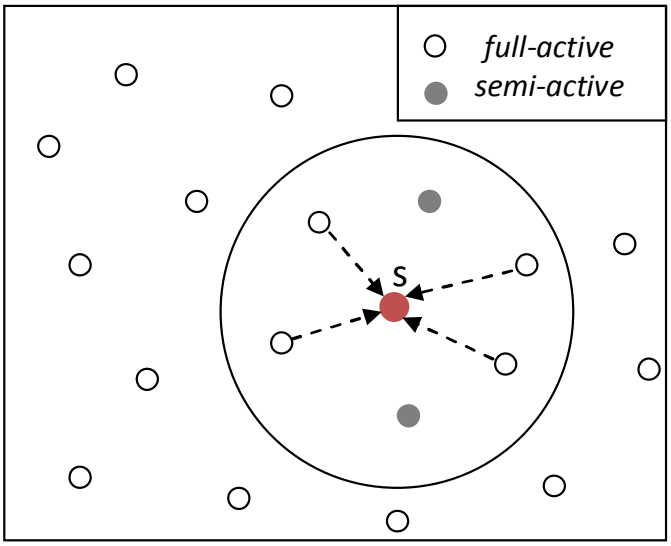

Figure 1. Limitation illustration of the current neighbor discovery.

ADAPCODE+

Here, we first introduce the basic assumption in our code distribution system, and then, we show the overall architecture of our new scheme, the AdapCode+, for the discovery of neighbor nodes followed by the detail explanation of the proposed scheme.

Basic assumption

In our protocol, we assume that sensor nodes are randomly distributed and they are not previously configured with the knowledge of their locations. In addition, there are *n* data messages, each with fixed length that can fit into a packet. There is one single source in the system. The source will keep on transmitting the code update packets and will pause for a period of T_p milliseconds after finishing a page. This pause of source is necessary to allow other nodes to start to propagate previous pages. The choice of T_p is a tradeoff between traffic and propagation time. All the other nodes will help spread messages they received. Those nodes will use network coding to minimize the number of transmissions while ensuring that every node in the system will correctly receive the updated code information.

Overall system architecture of AdapCode+

Figure 2 illustrates the code distribution process in our AdapCode+ scheme. The main idea of AdapCode+ is to

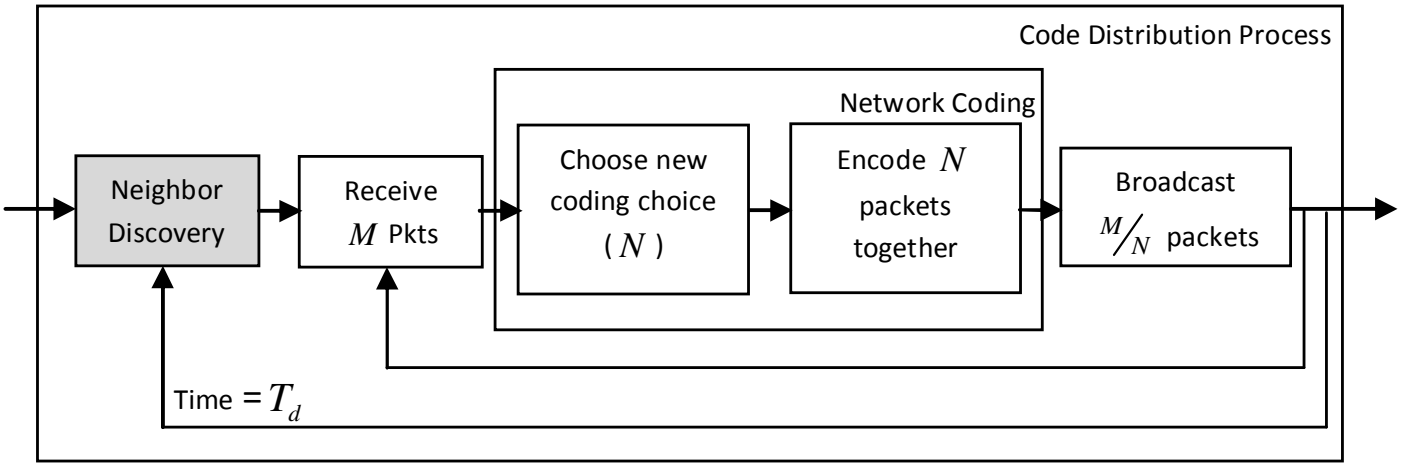

Figure 2. The code distribution process in AdapCode+.

Page#	Coefficient Vector	Data Segment

Figure 3. The packet format used in data broadcasting

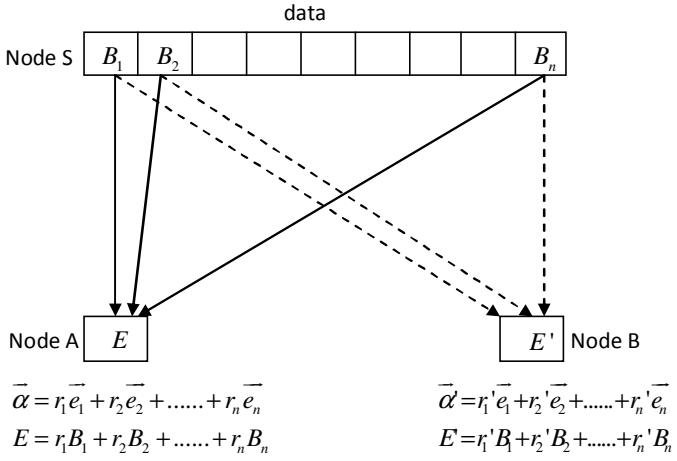

Figure 4. Data encoding at the relay node.

use the same network coding algorithm for receiving and broadcasting the packets used in AdapCode. However, to make sure that all the neighbor (both semi-active and full-active neighbor) nodes are detected, we deploy power efficient neighbor discovery phase in the beginning of code distribution process. When receiving the correct number of encoded packets, each node will decode the original packet. After that, the node will choose new coding choice depending on its number of neighbors and then broadcast the encoded packet. This process will repeat until it finishes the code distribution process.

Network coding algorithm

When a node receives a packet, it first runs Gaussian elimination to see if it has gathered enough information to decode all messages in the packet's page. When it succeeds in decoding all messages within a page, it determines its new coding choice, N, according to the number of its neighbors. When transmitting the packet, for each data segment, $B_i\ (i = 1,....,n)$ and its coefficient vector, e_i , the relay node first produces the *n* random numbers, $r_1, r_2, \ldots, r_n$, and then computes the new data segment E and coefficient vector $\vec{\alpha}$ by the following equations:

$$\vec{\alpha} = r_1\vec{e_1} + r_2\vec{e_2} + ... + r_n\vec{e_n} \qquad (1)$$

$$E = r_1B_1 + r_2B_2 + ... + r_nB_n \qquad (2)$$

Finally, the relay node transmits the encoded packet with the format as shown in Figure 3. We can see an example in Figure 4.

Thus, once the sink node has received at least N packets and the coefficient vectors of which are linearly independent, these N original packets can be recovered by using Gaussian elimination as we can see in Figure 5. Typically, Gaussian elimination requires two NxN matrixes, one to store the original coefficients and the other to store the inverse matrix. Thus, the total memory usage of Gaussian elimination would be less than 1k, which can fit in the memory of the modern sensor nodes.

In this paper, we consider linear network coding on finite Galois Field GF (2m) where m = 1. We choose the

$$\begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_n \end{bmatrix} = \begin{pmatrix} r_1^1 \ldots . r_n^1 \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ r_1^n \ldots . r_n^n \end{pmatrix} . \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ B_n \end{bmatrix} \Leftrightarrow \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ B_n \end{bmatrix} = \begin{pmatrix} r_1^1 \ldots . r_n^1 \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ r_1^n \ldots . r_n^n \end{pmatrix} . \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_n \end{bmatrix}$$

Figure 5. Original data recovery.

smallest possible Galois Field, GF (2) to decrease the computational complexity of coding operations. Each node generates a new packet which is a linear combination of the earlier received packets by coefficients in the finite field GF (2). When working in GF (2), the number of operations needed for encoding and decoding is the same. The sink can successfully recreate the original data packets after receiving N linear independent coded packets by using Gaussian elimination.

Neighbor discovery algorithm

Initially, in the self-organizing WSNs, nodes do not have the deterministic knowledge of the location of their neighbor nodes. Thus, they must transmit wireless queries in order to discover the neighboring nodes and establish a communication network. Neighbor discovery is the first application that runs on each sensor node to form a sensor network and it is also essential to find the updated network information for each time interval T_d. As mentioned above, we determine the coding scheme by the number of neighbors that a sensor has. The value of T_d should be determined according to the mobility of the network. For example, if nodes in the network move quite frequently, we should have a small value of T_d to be resilient against topology change. On the contrary, if the topology remains quite static over time, we should set T_d to a larger value to obtain a more accurate number. The sensor then decides the new coding scheme according to the number of neighbor nodes (Table 1).

In AdapCode+, neighbor nodes are discovered, thanks to the use of the network beacons. The utilization of the beacons for distributing the neighbor information has several benefits. First, beacon exchanges are synchronized with a MAC protocol. This provides very low idle listening and power overhead due to additional control signal exchanges. Second, beacons are quite short, which provides power-efficient implementation. In addition, there is no need for a new frame type for beacon messages. These benefits make the beacon very important for wireless networks especially for self-organizing networks.

In the initial neighbor discovery process, each sensor node broadcasts a specific number of beacon messages to advertise its presence. Due to the broadcast nature of radio communication, each sufficiently closed neighbor (both full-active and semi-active neighbors) node receives this beacon message and may infer that it is a neighbor of the sender. During this period, each node will constantly sample received signal strength (rss), and at the end of this period, all neighboring sensors will calculate the mean of the measured signal strength (q) and store it for later use. The use of this value, q, prevents a certain number of packet lost and thus affect the reliability of the network.

Each node maintains a local neighborhood table (NHTable), as shown in Figure 6. During the process of neighbor discovery, the identity of the discovered node whose signal strength is higher than q is added to the local neighborhood tables of the nodes which have received the beacon messages. The algorithm requires data memory for storing next-hop neighborhood information. As the number of neighbors is, at most, k, the total number of neighbor entries is upper bounded by k. Thus, the required program memory space for the algorithm is typically less than 1*kB* and hence it is suitable for the memory of the sensor nodes.

The neighbor discovery algorithm discussed can be

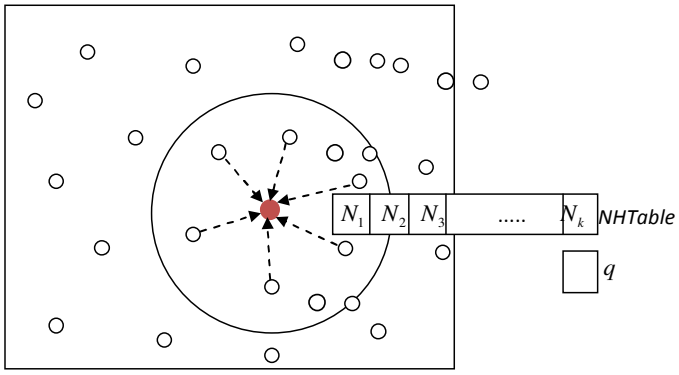

Figure 6. Neighbor discovery by using network beacon.

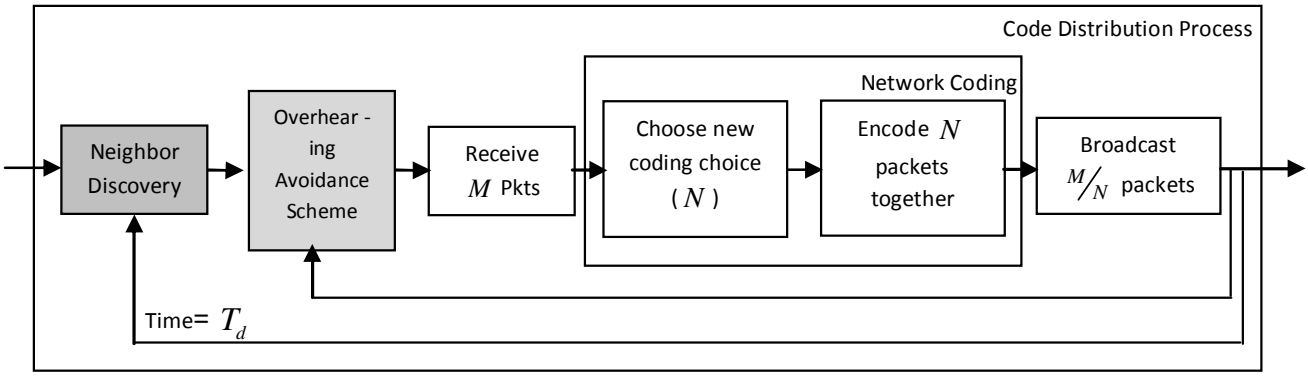

Figure 7. The code distribution process in AdapCode++.

summarized as follows:

Neighbor discovery algorithm

for each node ∈ network **do**

receive beacons from all nodes ∈ its communication area

if reception was successful from node i **then**

measure $rss(i)$ based on the beacon reception

if $rss(i) \geq q$ **then**

add node i to its local *NHTable*

end if

end if

update the mean of measured signal strength (q)

end for

AdapCode++

Although there is significant power savings achieved by the AdapCode+ from using neighbor discovery protocol, it still suffers the problem of overhearing from receiving redundant messages. In addition, due to its always onradio structure, it will also suffer the problem of idle listening (listening to an idle channel in order to receive possible traffic). To address the above problems, we further enhance the AdapCode+ to AdapCode++.

Overall system architecture of AdapCode++

The overall system architecture of AdapCode++ is shown in Figure 7. To further reduce the useless power dissipation for overhearing and idle listening, in AdapCode++, we propose an overhearing avoidance scheme that eliminates idle listening and the reception of useless redundant information by using the digest information of the subsequent data packets. This field is envisaged to make it possible for a node to identify and avoid receiving irrelevant broadcast frames before the entire reception. This is very useful for many wireless networks which use flooding to propagate information throughout the network, and thus, a node may receive multiple copies of the same frame contents from all its

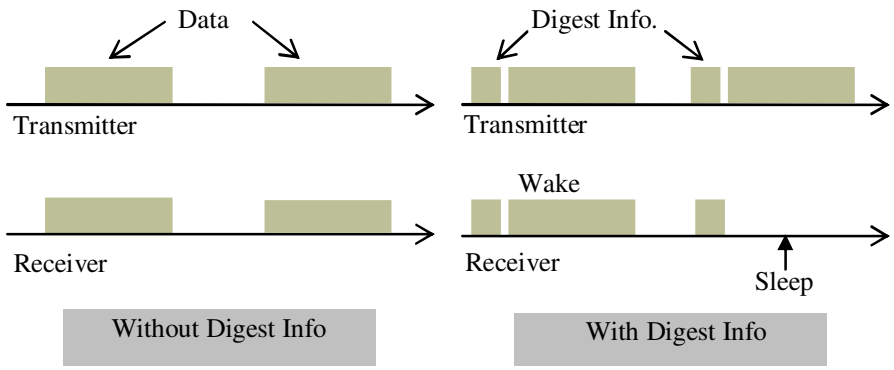

Figure 8. Avoiding overhearing of redundant message by means of 'Digest Info'.

neighbors.

Sleep-wake scheduling

In the wakeup interval of duty-cycling MAC protocols, if a node finds the channel to receive some messages, it continues listening to the messages, otherwise, it goes back to sleep immediately. In such protocols, unnecessary power drain happens as the node accepts to receive the messages without considering whether they are relevant messages to it or not. Therefore, we propose here a new 'digest info' packet to identify if the subsequent message is already received before by the node or not. In AdapCode++, a node will switch its radio on only when it receives a wakeup message from the source node.

After receiving a wakeup message, it first checks the 'digest info' field which is embedded in the wakeup message to determine whether it should receive the subsequent message or not. If it is the correct message for itself, it keeps its radio on until it receives the subsequent data message as shown in Figure 8. Otherwise, it immediately goes back to sleep mode again. And also, right after the reception of the subsequent

$$coefMatrix \leftarrow MxM\ matrix$$
$$invMatrix \leftarrow MxM\ matrix$$
$$hashTable \leftarrow n\ matrix$$

relevant data message, the node will go back to sleep mode.

The 'digest info' packet

The 'digest info' is a small frame embedded in the wakeup message and sent right before each data broadcast message. The 'digest info' packet has a field that contains either a unique identifier, or a hash of the data contained in the subsequent broadcast frame. The node uses the information in the 'digest info' field to learn about the subsequent data contents. If a node learns from the 'digest info' field that the following data packet has already been received, it can switch its radio off, because the subsequent data is redundant. In this way, the node overhears only 'digest info' packet instead of overhearing redundant whole data packet, which contributes to save much power since the 'digest info' packets are far shorter than the data packets.

When a node receives a wakeup message, it first checks its table whether there is an entry with the same hash value. If the entry exists, the node switches its radio off to avoid receiving the same data again. If there is no such entry in the table, the node keeps its radio on and continues listening to the channel in order to receive the subsequent data packet. Once it has received the data packet, the node inserts the hash value of the current received data packet to update its table in order to avoid receiving redundant transmissions. This table logs packets that have been recently seen so the MAC layer may switch the radio off when it expects a redundant reception. Figure 9 illustrates the flowchart of packet reception with the avoidance of overhearing. When a node needs to transmit a wakeup message to awaken the neighboring sleep nodes, it constructs and embeds the corresponding 'digest info' of the subsequent packet into the wakeup message and transmits it ahead to broadcasting the data packets. This process can be seen in Figure 10a.

According to this procedure, the MAC layer always receives the 'digest info' packet before a data packet for each broadcast communication. The digest information of the data frame significantly minimizes overhearing and thus increases the power saving of frame preamble MAC protocols. As only the first data content is useful, the node saves significant power amount by ignoring the subsequent broadcasts carrying the same data contents. Our new network coding algorithm can be clearly summarized as follows:

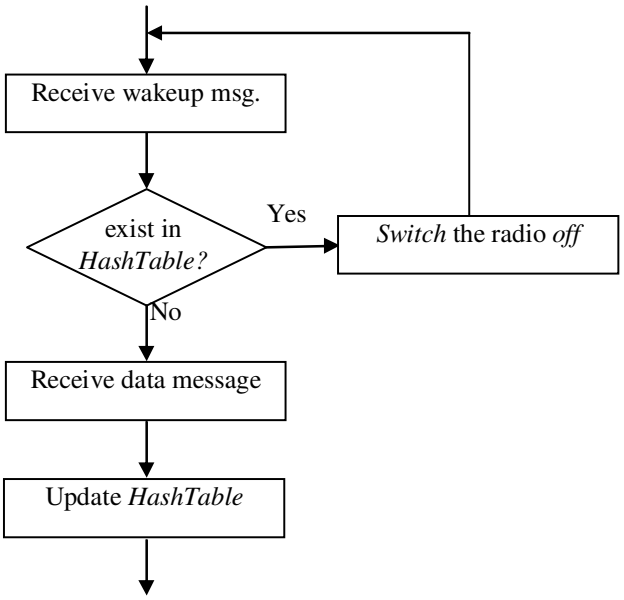

Figure 9. The flowchart of packet reception with overhearing avoidance.

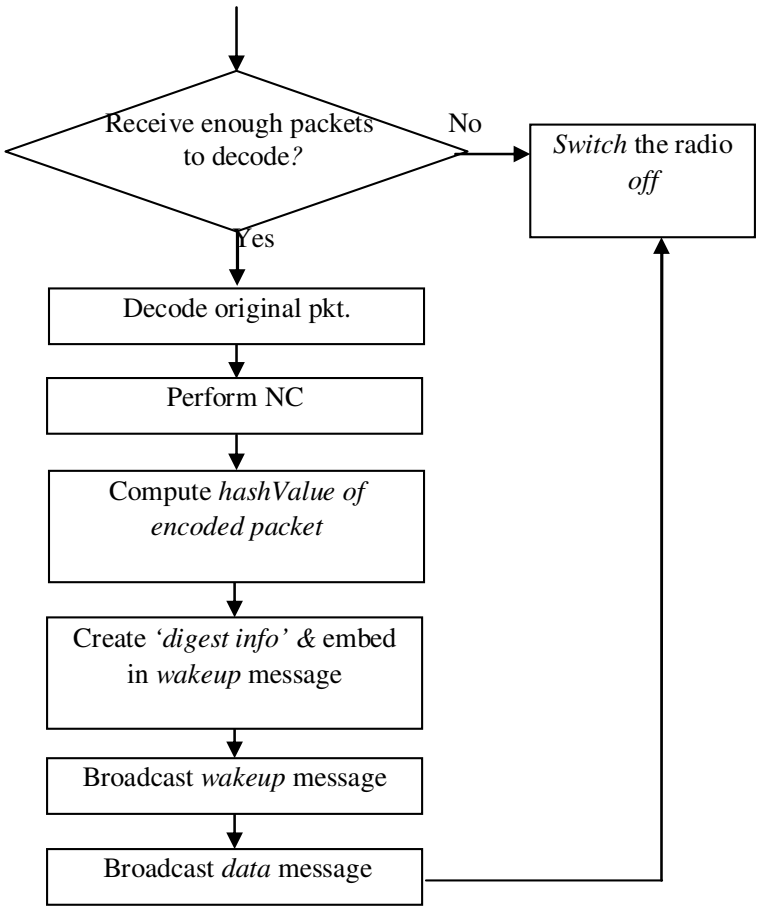

Figure 10a. The flowchart of packet transmission.

while code distribution is going on **do**
for each time interval T_d
$numNeighbor \leftarrow neighborDiscovery();$
if a wakeup message is received **then**
Check the $hashValue$ in 'digest info' packet
if same value in its $hashValue$ **then**
Switch its radio off
else
Receive the subsequent data message
Update its $hashValue$
$rank \leftarrow Gaussian(coefMatrix, invMatrix);$
if $rank = M$ **then**
Solve all messages in the page by $invMatrix$
Determine N according to $numNeighbor$
for $i = 1$ **to** $i \leq M/N$ **do**
Produce N $coefMatrix$
Compute linear combination of N messages
Compute $hashValue$ of encoded message and create 'digest info' packet
Embed 'digest info' in wakeup message and broadcast it
Broadcast data packets
end for
end if
Switch its radio off
end if
end if
end while

Figure 10b. Network coding algorithm.

Table 2. Simulation parameters.

Number of nodes	100
Area (m^2)	100x100
Mobility model	random waypoint
Pause time (s)	30
Max speed (m/s)	0-20
Channel capacity	$1Mb/s$
Data rate	4 packets/s
Transmission range	25 m

Network coding algorithm

This is shown in Figure 10b.

PERFORMANCE EVALUATION

Here, we investigate how much power efficiency can be further improved by adopting the proposed neighbor discovery algorithm and overhearing avoidance coding algorithm in AdapCode, as compared with the original AdapCode.

Simulation setting

We perform our experimental exploration using NS-2, a standard simulation tool in sensor network. We have implemented a simple code distribution application in which a single source node distributes the code update information and other nodes in the network forward the packets they received to spread out the updated code information throughout the network. The parameter setting for the simulation is shown in Table 2. For the sake of simplicity, we assume MAC protocol assigns a unique channel for every node to prevent possible collisions.

Multi-channel communication protocols (Cordeiro and Challapali, 2007; Son et al., 2007; Pollini, 1994), has been extensively used in WSN to increase system

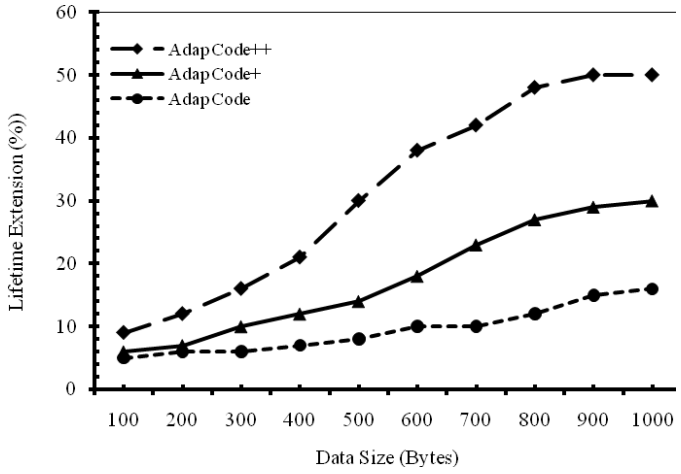

Figure 11. Lifetime extension with respect to the data size.

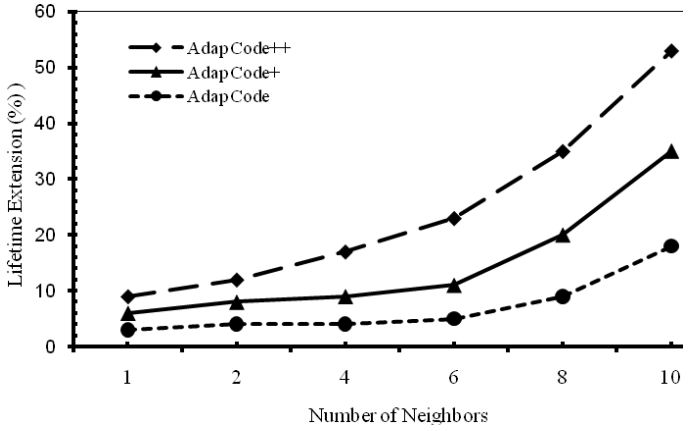

Figure 12. Lifetime extension with respect to the number of neighbors.

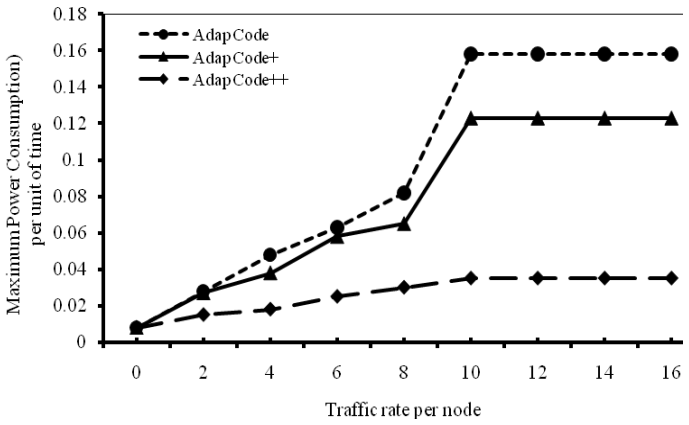

Figure 13. Power consumption.

throughput. Once different channels are assigned to interfering or contending links, more simultaneous transmissions can take place and thus the use of multi-channel MAC protocols mitigate the possible collisions. Moreover, this assumption is wide-ranging in many sensor network simulations for the simplicity.

Simulation result

We use NS-2 to quantify the lifetime extension achieved with the use of our proposed mechanisms by simulation. One can come up with various lifetime definitions for sensor networks in the literature. Here, in our discussion, we define network lifetime as the time until the first sensor node death occurs (FND). We compare the lifetime achieved by three protocols: the AdapCode, available network coding architecture; the AdapCode+, our first improvement over AdapCode with the neighbor discovery method; and the AdapCode++, our further improvement over AdapCode+ by using the 'digest info' packet with the Deluge as baseline protocol. We carry out simulations to get more insights into the power saving ratio since we have not taken all the parameters into account in the mathematical analysis. For each simulation run, we calculate the lifetime extension as defined above. After each 10 simulation runs, we calculate the average lifetime extension. Figure 11 shows the lifetime extension we achieved for various data packet sizes. According to the simulation results, we can conclude that the lifetime extension is small for small data sizes, because the amount of time during which we switch the radio off to avoid redundant data reception becomes negligible compared to the time the radio is on. However, when the data payload size increases to 512 bytes, the lifetime extension also increases by 40 to 50%.

In Figure 12, we have measured the life time extension ratios for various numbers of neighbor nodes. In here, the number of neighbors determines the density of the network. As we expected, the lifetime extension increases when the density of the network increases. Although the gain is not much for the sparse networks, we get the significant power savings for the dense networks. We can easily see from Figure 12 that the lifetime extension significantly increases during the large number of neighbors (that is, $numNeighbor \geq 4$).

Figure 13 plots the power consumption of each node for each different traffic rate. Note that, even though the power consumption is not much different for low traffic rate, our protocols can sustain for high traffic rate. The AdapCode+ can also reduce the power consumption but the AdapCode++ outperforms the performance of AdapCode+ since the AdapCode++ includes less idle listening time and overhearing avoidance scheme. For example, at the traffic rate of 6 bytes/s, the AdapCode+ consumes the power of 0.06 while the AdapCode++

consumes about 0.025. We can also see from Figure 13 that, starting from the traffic rate of 10 bytes/node/s, the power consumption of AdapCode+ drops with a factor of 0.12 while the AdapCode++ drops with a factor of 0.04.

CONCLUSION

Power efficiency is a key issue for WSNs, since sensor nodes are powered by non renewable batteries. Our work is different from the available AdapCode which is by now, the promising network coding architecture in terms of power saving in WSNs. First, we proposed AdapCode+ in which we use the neighbor discovery protocol for getting more benefits in network coding process, and the AdapCode+ achieves a certain power saving than the AdapCode. In addition, we further enhanced AdapCode+ to AdapCode++ by proposing a new idea for power saving to reduce idle listening and avoid overhearing redundant copies of broadcast messages. With the use of digest information of the subsequent data message, our proposed coding architecture, the AdapCode++, promises same reliability with the AdapCode while consuming significantly less amount of power. We also have evaluated our proposed schemes analytically and by means of simulation in NS-2. As a main contribution of our paper, we show that, significant power savings can be achieved by efficiently reducing the unnecessary redundant packet receptions compared to the simple network-coding based protocols.

REFERENCES

Akyildiz F, Su W, Sankarasubramaniam Y (2002). Wireless Sensor Networks: a survey. Comput. Networks, 38: 393-422.

Cordeiro C, Challapali K (2007). C-MAC: A Cognitive MAC Protocol for Multi-Channel Wireless Networks. IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks. 147-157.

Demirkol J, Ersoy C, Alagoz F (2006). MAC Protocols for Wireless Sensor Networks: a Survey," IEEE Trans. On Comm. 44: 115-121.

Fragouli C, Wismwer J, Boudec JYL (2006). A Network Coding Approach to Energy Efficient Broadcasting: from Theory to Practice. ACM MobiSys.

Haenggi M (2004). Opportunities and Challenges in Wireless Sensor Networks. Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems. CRC Press. Florida. 1.1-1.14.

Polastre J, Hill J, Culler D (2004). Versatile Low Power Media Access for Wireless Sensor Networks. SenSys., 11: 95-107.

Pollini GP (1994). The tree-search resource auction multiple access (TRAMA) protocol for wireless personal communications. Proceedings of IEEE Vehicular Technology Conference (VTC). pp. 1170-1174.

Son C, Lee NH, Kim B, Bahk S (2007). MAC Protocol Using Asynchronous Multi-Channels in Ad Hoc Networks. IEEE Wireless Commun. Networking Conference, pp. 401-405.