An energy-efficient command broadcast protocol in dynamic WSNs

Ta Van Khoe*, Phan Trong Hanh

Faculty of Radio-Electronic Engineering, Le Quy Don Technical University.
*Corresponding author: tavankhoe@gmail.com
Received 01 Dec. 2022; Revised 13 Feb. 2023; Accepted 10 May 2023; Published 25 May 2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.87.2023.20-31

ABSTRACT

In wireless sensor networks (WSNs), a sink must broadcast periodically a command message to a group of nodes or all nodes to control the operation of the network. However, the sparse distribution and limited resources of nodes bring difficulties for efficient broadcasting. In this paper, we propose an energy-efficient command broadcast protocol based on the construction of a spanning tree which includes maximizing the number of leaf nodes, and the allocation of one sharable slot to each tree level. The sharable slot that allocates to tree level will be divided into multiple mini slots, and non-leaf nodes at the same level will compete for broadcasting; in this way, collision can be reduced. Moreover, a node that moves to another level can use the sharable slot allocated to the new level without reallocating the slots. Extensive simulations prove that this approach not only achieves high reliability of message delivery but also energy efficiency.

Keywords: Command broadcast; Leaf nodes; Sharable slot scheduling; Node mobility; Energy efficiency.

1. INTRODUCTION

A wireless sensor network (WSN) includes a number of sensor nodes that collect data about the ambient environment and propagate it to the server (sink). A server is required to send a command message to a group of nodes or all nodes for monitoring and controlling the network operation. The message needs to be delivered in an energy-efficient, timely, and reliable manner. Especially, energy efficiency is a critical issue since the replacement of batteries is often cumbersome and time-consuming. However, it is not easy to achieve the requirements because of node movement, internal and external interference, and diverse obstructions [1]. Based on the method in which a node can retransmit control messages, broadcast protocols can be divided into three categories. 1) flooding [2-5]: a node will rebroadcast a received message to its neighbors as soon as it receives the message for the first time or with its probability. 2) selective broadcasting [6-8]: selective nodes will rebroadcast a command message in contention or schedule-based transmission. 3) slotted broadcasting [9, 10]: intermediate nodes will retransmit a received message in a distinct slot.

In the first category, the Flooding protocol [2] achieves high reliability of packer delivery; however, it suffers from the storm problem [11]. In the Glossy protocol [3], every node will rebroadcast the receiver message immediately and deal with a collision by utilizing the notion of constructive interference. However, this approach may still suffer from considerable energy consumption since nodes perform a lot of unnecessary rebroadcasts. Other while, in [4, 5] a node decides where to rebroadcast or not based on a probability function to suppress unnecessary dissemination. In [4], a node gets a higher rebroadcast probability in sparse networks while it gets the lower one in dense ones. In DPFNI [5], a node calculates its rebroadcast probability more precisely according to two hop neighbor information, such that the more siblings it has, the less rebroadcast probability it gets, and the more children it has, the higher rebroadcast probability it gets.

In the second category, approach in [6] constructs a *connected dominating set* (CDS) and the flooding is done if and only if nodes are in the constructed CDS. In [7], the authors reduce collision of flooding by assigning distinct slots to the nodes in CDS. However, finding a minimal CDS is known to be NP-complete [12]. In the third category, the RSBP [9] is the broadcast protocol for timely message transmission with low energy consumption. In which, a sink allocates a distinct slot to each node except for leaf nodes so that every node can rebroadcast a message in a contention-free manner. The approach gives a predictable end-to-end delay of a message as well as an efficient energy management; however, every node has only one chance of receiving a broadcast message. The authors in [13] prove that the decrease in number of forwarding nodes by increasing a number of leaf nodes in a spanning tree is an energy-efficient broadcast method. A leaf node in the tree needs to turn on itself for only to periodically sense and receive command message from its parent and the node can remain asleep for the rest of the time.

In this paper, we propose a novel broadcast protocol that combines the strengths of flooding, slotted broadcasting, and selective broadcasting. In the approach, the tree construction phase will optimize leaf nodes level by level in a distributed manner, enabling each node to select a suitable parent while not increasing the tree level. Then, the approach utilizes a sharable slot from the slotted sense multiple access broadcast protocol [10] such that one distinct sharable slot is allocated to each tree level and only the nodes at the same tree level compete for rebroadcasting.

The rest of this paper is organized as follows. In section 2, we give the preliminaries. Then, we present a formal description of the proposed protocol in section 3, followed by section 4 which evaluates the proposed protocol comparatively with related approaches. Finally, we give concluding remarks in section 5.

2. PRELIMINARIES

2.1. Network model

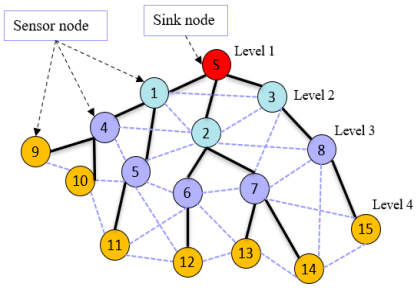


Figure 1. *Network model with spanning tree.*

A considered WSN consists of a number of *sensor nodes* which are battery-powered and one *server* or *sink* which is wall-powered. The nodes can form a multi-hop tree originating from the sink. Two nodes that can communicate directly with each other are said to have a *link*. A link can be broken due to node failure, battery depletion, node movement, or the intervention of some obstacles, thereby causing network topology to be changed dynamically. If a node belongs to a tree, it is said to be a *tree-node*; otherwise, it is an *orphan-node*. Especially, a link between a parent and its child is called

a *tree-link*. The Fig. 1 shows a tree topology that consists of sixteen nodes which are tree-nodes, and a tree-link and an ordinary link are denoted by a solid line and a dashed line, respectively.

2.2. Motivation

The features of a broadcast protocol and their effects on the quality metrics including reliability of message delivery (RMD), energy consumption (EC), and end-to-end delay of message delivery (E2ED) are shown in Fig. 2. Firstly, using overlapped transmission can shorten the E2ED. It will, however, increase collision, leading to high EC and low RMD. The constructive interference idea, which enables a receiver node to successfully demodulate numerous concurrent signals, can be used to overcome this issue. Secondly, the use of random delay and channel checking before rebroadcasting will increase the RMD by reducing the possibility of collision and allowing nodes a chance to receive the same message multiple times. However, this method will increase the waiting time and duty cycle, leading to high E2ED. It also suffers from high EC due to the increased competition from message transmission. Thirdly, the channel and slot scheduling approach will remove channel completion and lessen conflict based on allocating a distinct slot and channel to each node, resulting in high RMD. Additionally, the approach reduces a node's active time, which results in energy efficiency. However, the assignment of a distinct slot to each node will lengthen the super frame size, which is the time for every node to broadcast at once, leading to an increase in E2ED. Finally, the use of selective nodes will reduce EC and E2ED by reducing the number of message transmissions and receptions. However, it will lower the RMD.

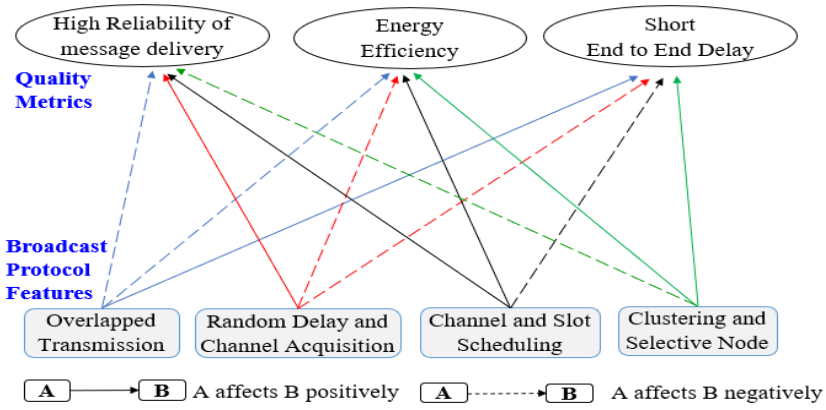


Figure 2. *Broadcast protocol features and their effects on quality metrics.*

In the paper, we propose *an energy-efficient command broadcast protocol* (EECBP) for dynamic networks by maximizing *leaf nodes* and employing *sharable slots*. In the approach, a reliable tree is constructed, and a node selects its parent node which has the maximum number of children while not increasing the tree level to maximize number of leaf nodes level by level. Then, nodes at the same tree level compete for broadcasting within a sharable slot using a random delay thus generating a topology-independent slot schedule and lead to the meaningful reduction of collision. A node has multiple chances of receiving the command message from the nodes at one level lower. If a node does not receive broadcast message, it will wait continuously in *addition sharable slots* to receive broadcast message from the nodes at the same and one higher level. This contributes to

increasing the reliability of message delivery. A sharable slot will be further divided into multiple mini-slots and the slotted Aloha approach [14] can be borrowed to reduce the possibility of collision. Every node tries to broadcast a message only at the beginning of a mini-slot. If a node overhears the broadcast message from one of its neighbors, it delays its broadcast time to one of the later mini-slots within its own sharable slot.

The advantages of the EECBP are several. First, it can save energy consumption and reduce collision by reducing broadcasting nodes and competing with intermediate nodes. Second, it can respond well and quickly to the change in topology. Since when a node moves to another level, it needs only to change its sharable slot to the one allocated to the changed level. Third, the approach enables multiple nodes to perform parallel transmission if they are distanced properly. Lastly, it can achieve relatively high reliability of message reception since nodes often have connections to multiple nodes.

3. EECBP PROTOCOL

3.1. Protocol structure

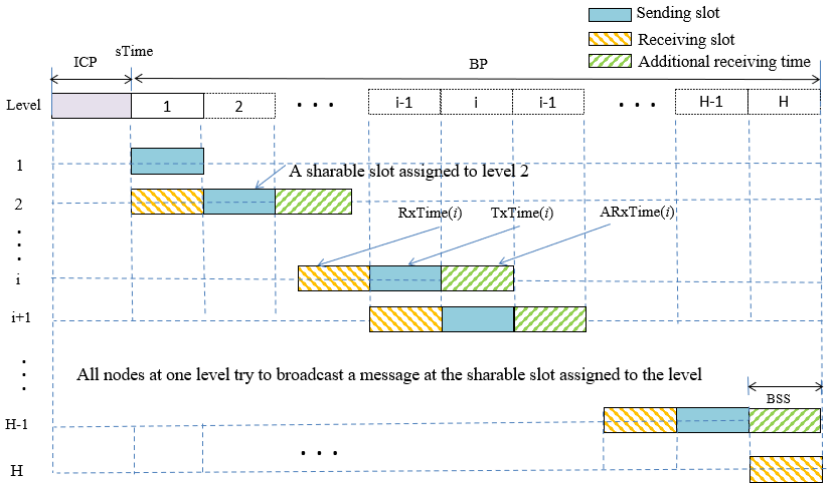


Figure 3. *Protocol structure and the allocation of a broadcast sharable slot (BSS).*

In our approach, the protocol includes the *initial construction period* (*ICP*) at starting time and repeats the *broadcast period* (*BP*) continuously as shown in Fig. 3. During ICP, tree establishment, time synchronization, and slot scheduling are performed sequentially. BP starts with a globally synchronized time, *sTime*. During BP, the broadcasting is performed, starting with a sink at the lowest level and proceeding to the nodes at the next level progressively. Given a tree of depth *H*, BP is divided into *(H-1) broadcast sharable slots* (*BSS*) since nodes at level *H* do not have to rebroadcast the received message. During the i^{th} *sending slot* allocated to level *i*, only the intermediate node(s) at level *i* are in the broadcasting mode and each node goes to sleep immediately after rebroadcasting while leaf node(s) go into sleep mode instantly upon receiving message. At the i^{th} *receiving slot*, the node(s) at level *i+1* are in the receiving mode and each node goes to sleep immediately after receiving. If those do not receive message in its *receiving slot*, it continuously stays in the receiving mode in an *additional receiving time* to get message from neighbor at the same level and higher level, respectively.

3.2. Tree construction and time synchronization

3.2.1. Tree construction

A link quality metric ($linkq$) that combines two physical link quality metrics of the *Received Signal Strength Indicator* (RSSI) [15] with the *Link Quality Indication* (LQI) [16] can be used to judge a link as reliable or not. However, a link may not be bi-directionally reliable link (*BR-link*) or its quality may vary according to the strength of transmission powers and the effects of path loss, shadowing, fading, etc. Hence, we propose a new efficient approach to identify the *BR-link* base on exchanging control messages. The detail of reliable tree construction by using control messages such that link is *BR-link* and the number of the leaf nodes are maximum as follows.

A sink initiates tree construction process by broadcasting a *Tree Construction Request* (*TCR*) message. Upon receiving *TCR*, an *orphan-node, u*, joins the sink by sending a *Join Request* (*J-REQ*) message. Upon receiving *J-REQ*, the sink sends a Join Response (*J-RES*) message and takes the *orphan-node* as its child. When node *u* receives *J-RES*, it takes the sink as its *primary parent*. Another *orphan-node, i*, that has overheard *J-REQ* for the first time from node *u* sets its level *level*(*i*) to *level*(*u*)+1 and initiates a waiting time for receiving more *J-REQ's* from other nodes. During the waiting time, node *i* maintains a parent candidate list, *PCL*(*i*) upon overhearing J-REQ's. To prevent collision of the J-REQ message, each node sets a *joint delay timer*, (*jDelay)*, as follows before sending J-REQ message.

$$jDelay = D \times (level(Re) - level(Tx) + r) \quad (1)$$

where *D* is a constant delay for single hop forwarding; $level(Re)$ and $level(Tx)$ are tree levels of an receiving node and a sending node of TCR or J-REQ, respectively ($level(sink)$ = 1); *r* is a random number in [0,1].

When the *jDelay* expires, node *i* selects a *tree-node* in *PCL*(*i*) which has the biggest number of children as a *primary parent* to join. Information about the number of children that a node has is constructed based on overhearing *J-RES*. Whenever node *i* overhears a *J-RES* sent from node $v \in PCL(i)$, it increases the *children counter* by 1 for node *v*. This process repeats until there is no *orphan-node* in the network. We detail the tree construction process in Algorithm 1.

Algorithm 1. *Tree construction algorithm to maximum number of leaf nodes.*

Event: node *i* receives *TCR* from a sink
1: level$(i) \leftarrow 2$;
2: $delay \leftarrow generateRandomDelay();$
3: send *J-REQ*(sink, *i*, level(i));

Event: node *i* receive *J-REQ* from node *u*
1: send *J-RES*(*i*, *u*);
2: $Children\_List(i) \leftarrow$ node u;

Event: node *i* overhears *J-REQ* from node *u*
1: **if** $((u \in PCL(i) == true)$ **then**
2: $children\_counter(u) \leftarrow child\_counter(u) + 1$;
3: **end if**

Event: the *delay_timer* expires at node *i*
1: $p \leftarrow \emptyset$; $maxNumOfChild = 0$;
2: **for** $(v \in PCL(i))$

node u:
1: **if** ((level(i) == 0) || (level(i) ≥ level(u) + 2)) **then**
2: level(i) = level(u) + 1;
3: $PCL(i) \leftarrow \emptyset$;
4: $PCL(i) \leftarrow u$;
5: **else if** (level(i) == level(u) + 1) **then**
6: $PCL(i) \leftarrow u$;
7: set *delay_timer*;
8: **end if**

3: **if** (max$NumOfChild \leq$ $child$ren_$counter(v)$) **then**
4: $p = v$;
5: max$NumOfChild = child_counter(v)$;
6: **end if**
7: **end for**
8: **if** ($p \neq \emptyset$) **then**
9: send *J-REQ* (p, i, level(i));
10: **end if**

3.2.2. Time synchronization

In the flooding time synchronization protocol (*FTSP*) [17], a sink floods a WSN with a synchronization message so that each node can receive it multiple times from its neighbors, and it then performs the linear regression to compensate for clock skew. It was proven by experiment that it could maintain an accuracy less than 30 microseconds for a resynchronization period of 30 seconds. In our approach, the *FTSP* protocol is used only once at the time of tree construction. Then time synchronization is performed gradually from the sink to the other nodes as follows. After every δ^{th} BP, a sink piggybacks the global time into a broadcast message and repeats this β consecutive times. This implies we do not need an additional time synchronization message. Upon receiving a broadcast message with the global time, it updates its local time and global time according to the process of the *FTSP* protocol.

3.3. BSS scheduling

The protocol structure is shown in Fig. 3, in which, the sharable slots are identical in size. The receiving BSS allocated to level *i* completely overlaps with the sending BSS allocated to level *i-1*, since the nodes have to receive a broadcast message while their parents broadcast the message. The approach protocol maximizes the number of leaf nodes level by level, so it reduces the chance of receiving broadcast message. To overcome this problem, if the node(s) at level *i* do not receive broadcast message when all the nodes at level *i-1* finish broadcasting, it continuously stays in receive mode in next BSS and additional receiving time to get message from neighbor at the same level *i* and neighbor at one higher level *i+1*, correspondingly. Accordingly, a receiving start time $R_xTime(x)$, a sending start time $T_xTime(x)$, and additional receiving start time $AR_xTime(x)$ of node *x* are expressed as follows:

$$\begin{aligned} &R_xTime(x) = sTime + (l_x - 2) \times Size(BSS), 2 \leq l_x \leq H \\ &T_xTime(x) = sTime + (l_x - 1) \times Size(BSS), 1 \leq l_x \leq H - 1 \\ &AR_xTime(x) = sTime + l_x \times Size(BSS) \end{aligned} \quad (2)$$

where l_x indicates the level of node *x*, *sTime* indicates a globally synchronized time, and $Size(BSS)$ indicates the *time length* of BSS.

3.4. Message broadcast

3.4.1. Collision avoidance method

Once the BP starts, every node determines its own BSS using Eq. (2) in a fully

distributed manner and remains sleep mode until it wakes up at its R_xTime or T_xTime. A sink, named *s*, starts broadcasting a message at time $T_xTime(s)$ while its child, say *x*, at level 2 waits to receive the message at $R_xTime(x)$. Every node goes to sleep as soon as it receives the message and wakes up at its T_xTime to broadcast the received message. The nodes at tree level *k* compete for broadcasting within the BSS allocated to its level. This section discusses the broadcast method that deals with the problem of message collision.

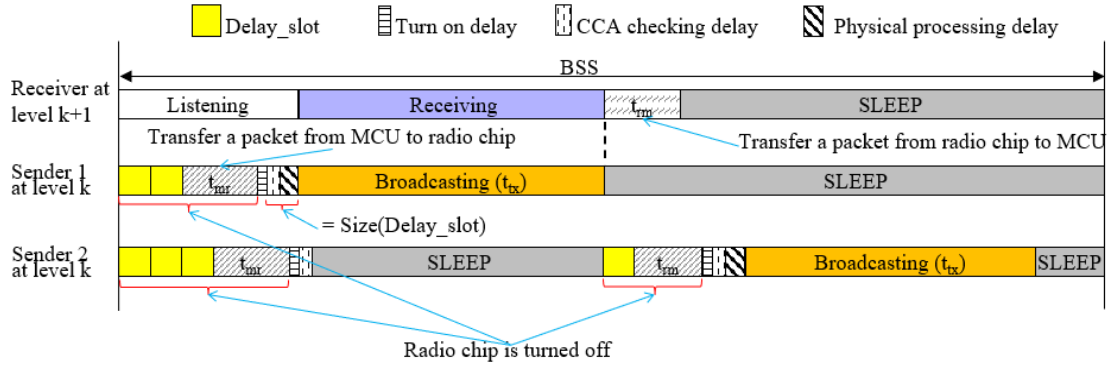


Figure 4. *Competition for channel contention.*

In a contention-based transmission (CBT) approach, the neighbor nodes at the same level compete to broadcast a message. A node takes the following message transmission process, as illustrated in Fig. 4. It transfers a message from MCU to radio chip and turns on its radio chip that checks the clear channel assessment (CCA). If the channel is idle, the radio chip takes the physical layer processing delay before it starts broadcasting. Therefore, two neighboring nodes have to get the difference of their random delay times by at least *delay_slot* that corresponds to the sum of a physical layer processing delay (= 12 symbols) and the CCA checking delay (= 8 symbols), according to the IEEE 802.15.4 physical layer standard [18]. If two neighbor nodes delay their broadcasts by the difference of at least one *delay_slot*, one node will never start transmitting a message before overhearing at least one bit transmitted earlier by its neighbor. Thus, every node uses the following random delay function, *rdelay*.

$$rdelay = random(0, CW) \times delay\_slot \tag{3}$$

where *CW* is a contention window. Thus, if two nodes generate different random numbers, one node with the bigger random number is delayed without causing collision at the same receiver.

Even though the *CBT* approach can reduce collision problem, it still suffers from the hidden terminal problem. A *slotted CBT* (*SCBT*) approach aims at alleviating the possibility of collision by employing the slotted Aloha method. A *BSS* is further divided into a number of *mini broadcast sharable slots* (*mBSS's*) as follows

$$BSS = (mBSS_1, mBSS_2, \dots, mBSS_N)$$

where, every *mBSS* is sufficient to broadcast one message and $N = \lfloor BSS/mBSS \rfloor$. Nodes contend a channel for broadcasting at the beginning of *mBSS* by using the delay function, *rdelay*, as illustrated in Fig. 5. Every node waits for *rdelay* and starts transmitting if it senses the idle channel. A node that fails to acquire the contended *mBSS* is delayed to the next *mBBS* for competition.

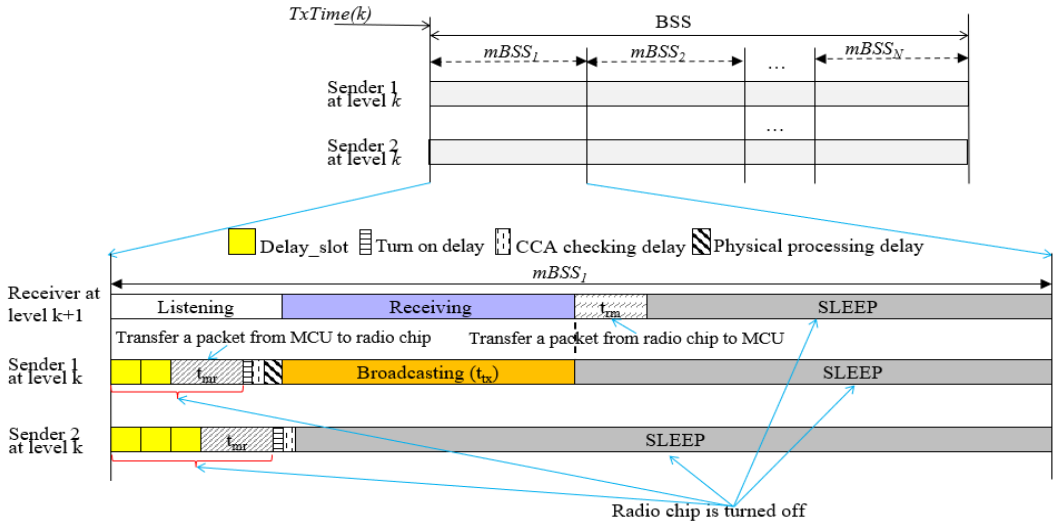


Figure 5. *Collision avoidance using mini broadcast sharable slots.*

However, even though this slotted approach can mitigate the possibility of collision, it can still suffer from collision if the competition for broadcasting within the same *mBSS* involves many neighboring nodes and/or any hidden node(s). Thus, it would be desirable to distribute the competition start times of the nodes over different *mBSS*'s. A node *x* can select its starting *mBSS* out of *N mBSS*'s using the following *start_mBSS*(*x*):

$$start\_mBSS(x) = random(1, N) \tag{4}$$

3.4.2. Discussion on key protocol parameters

The *mBSS* should be estimated to increase channel efficiency. Within *mBSS*, each node goes through the following seven steps to broadcast a message successfully: (1) set a random delay to avoid collision (*rdelay*); (2) transfer a message from the MCU to a radio chip buffer (t_{mr}); (3) turn the radio chip on (t_{on}); (4) check CCA (t_{CCA}); (5) set the physical layer processing delay (t_{ppd}); (6) transmit (or broadcast) a message (t_{tx}); and (7) transfer a message from the radio chip to the MCU at the receiver size (t_{rm}). The completion of each step needs some processing time or delay, and steps (1) to (6) take place on the sender side while step (7) is performed on the receiver side. Thus, one broadcast time, size(*mBSS*), is given as follows:

$$size(mBSS) \geq CW \times rdelay + t_{mr} + t_{on} + t_{CCA} + t_{ppd} + t_{tx} + t_{rm} \tag{5}$$

The *BSS* should be optimized to decrease the E2ED. The lower bound of *BSS* can be expressed as follows:

$$size(BSS) \geq size(mBSS) \times nCNodes \tag{6}$$

where *nCNodes* is the number of competing node and it can be estimated in [10].

4. PERFORMANCE EVALUATION

4.1. Simulation setup

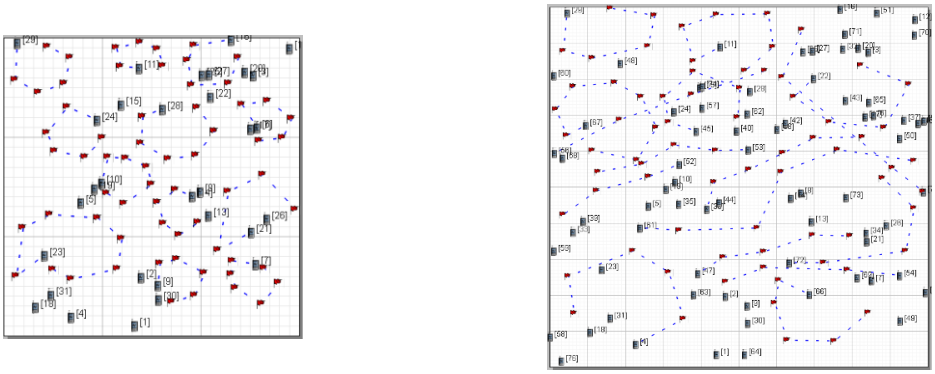
EECBP is compared with a Flooding protocol [2] and one slotted broadcast protocol

(RSBP) [9] in three metrics: (1) *Packet delivery ratio* (*PDR*) - the ratio of the number of nodes that receive the broadcast message to the total number of nodes, (2) p*acket processing load* (*PPL*) - the number of broadcast messages that each node receives and overhears and also transmits on average, and (3) *average energy consumption* (*AEC*) - the average amount of energy consumption per a node during the total simulation time. The evaluation of protocols is performed with varying percentage of mobile nodes (*Mnodes*).

Table 1. *Key simulation parameters and values.*

Parameters	**Scenario S1 Values**	**Scenario S2 Values**
Dimension	30×30 (m^2)	100×100 (m^2)
Number of Nodes	1 sink + 30 nodes	1 sink + 75 nodes
Transmission Range	10 m (-29 *dBm*)	28 m (-24 *dBm*)
Payload, *p*	20 bytes	
Rician fading (*dB*)	K = 8 (*dB*)	
Traffic model	Two-ray path loss model	
Node distribution	Random	
Mobility model	Random waypoint	
Mnodes (*%*)	10% to 40%	
Speed (*mps)*	Average 1.0 and max. 1.5	

The simulation was performed using QualNet version 5.0.2. A sink broadcasts one message whose MAC payload is 20 bytes every broadcast period of 0.5 seconds. The dynamic scenarios S1 and S2 are used for testing as shown in Fig. 6, and the key simulation parameters are summarized in table 1. Considering the short transmission range, the simulation was run for 400 cycles which corresponds to 200 seconds sufficient to make a topology change to a considerable extent. Note that the Rician fading model is employed with the Rician value, *K*, of 8 (*dB*). The *K* is defined as the ratio of the receiving power in the direct path (line of sight) to that in other paths. The random waypoint model is used with a maximum speed of 1.5 meters per second and a pause time of 5 seconds and *Mnodes* is changed from 10% to 40%.



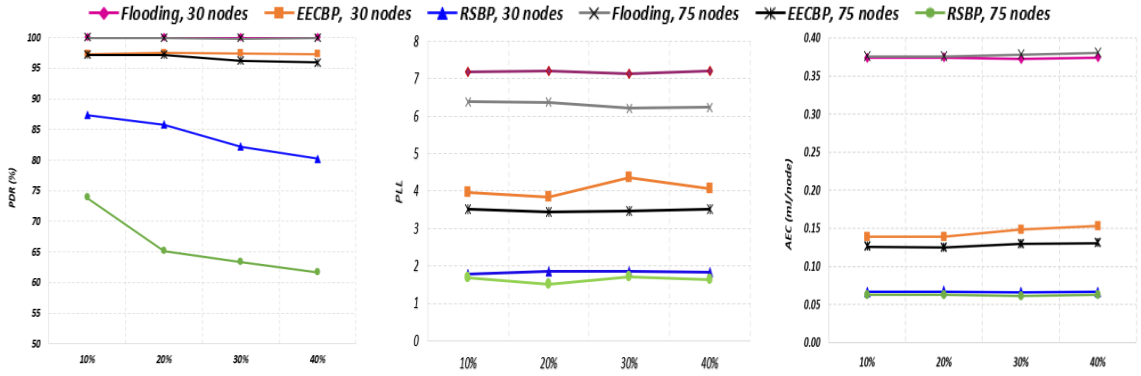
a. Scenario S1 *b. Scenario S2*

Figure 6. *Two network scenarios with mobile nodes.*

4.2. Discussion of simulation results

Fig. 7(a) shows that the reliability of RSBT is unreliable with the relatively high

Mnodes because the protocol relies on one small slot allocated to each broadcasting node. As expected, it is proven that RSBP does not respond well to node mobility. On the contrary, it shows that the Flooding is very reliable, and PDR of Flooding is not affected by the increase of *MNodes*. Meanwhile, the EECBP also shows a considerably reliable PDR even when *MNodes* = 40%, slightly lower than the Flooding protocol. The results indicate that both the Flooding and EECBP are very reliable against node mobility. The EECBP can achieve high reliability since a node has multiple chances of receiving the same message from multiple nodes at its parent level even though it broadcasts and receives or overhears less number of messages than Flooding as shown in Fig. 7(b). However, note that the energy consumption of EECBP is highly less than that of the Flooding and it is highly comparable to that of RSBP as shown in Fig. 7(c).



a. Packet delivery ratio *b. Packet processing load* *c. Average energy consumption*

Figure 7. *Comparison of different protocols in dynamic networks.*

5. CONCLUSIONS

We presented a novel energy-efficient and reliable broadcast protocol which makes use of a sharable slot for level after maximizing number of leaf nodes. First, the protocol can respond well to the change of topology and high signal interference because it uses a sharable slot. Second, it provides an energy-efficient management scheme that reduces number of intermediate node and allows a node to go into sleep or active mode effectively. Third, it generates low control overhead, since it performs tree construction and time synchronization only once. This paper showed intensive simulation results that EECBP far outperforms the other two broadcast protocols in terms of packet delivery ratio, average energy consumption, and packet processing load under dynamic network topologies.

REFERENCES

[1]. A. Sikora and V. F. Groza, *"Coexistence of IEEE802.15.4 with other Systems in the 2.4 GHz-ISM-Band,"* in 2005 IEEE Instrumentationand Measurement Technology Conference Proceedings, vol. 3, pp. 1786-1791, (2005), doi: 10.1109/IMTC.2005.1604479.

[2]. A. Hassanzadeh, R. Stoleru, and J. Chen, *"Efficient flooding in Wireless Sensor Networks secured with neighborhood keys,"* in 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 119-126, (2011), doi: 10.1109/WiMOB.2011.6085415.

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

[4]. Z. Qi and D. P. Agrawal, *"Dynamic probabilistic broadcasting in mobile ad hoc networks,"* in 2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484), vol. 5, pp. 2860-2864, (2003), doi: 10.1109/vetecf.2003.1286132.

[5]. H. Jeong and Y. Yoo, *"Dynamic probabilistic flooding algorithm based-on neighbor information in wireless sensor networks,"* in The International Conference on Information Network 2012, pp. 340-345, (2012), doi: 10.1109/icoin.2012.6164421.

[6]. W. P. Jun, K. M. Alzoubi, and O. Frieder, *"Distributed construction of connected dominating set in wireless ad hoc networks,"* in Proceedings of Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1597-1604, (2002), doi: 10.1109/infcom.2002.1019411.

[7]. A. FathimaRamzi and N. Sabiyath Fatima, *"Collision Optimized Broadcast Scheduling in Wireless Sensor Network,"* in International Journal of Computer Applications (0975 – 8887).

[8]. S. J. Mohammed and S. T. Hasson, *"Modeling and Simulation of Data Dissemination in VANET Based on a Clustering Approach,"* in 2022 International Conference on Computer Science and Software Engineering, pp. 54-59, (2022), doi: 10.1109/CSASE51777.2022.9759671.

[9]. P. Van Vinh and H. Oh, *"RSBP: A Reliable Slotted Broadcast Protocol in Wireless Sensor Networks,"* Sensors (Basel, Switzerland), vol. 12, pp. 14630-14646, (2012), doi: 10.3390/s121114630.

[10]. D.-S. Yoo, V. K. Ta, B.-T. Jang, and H. Oh, *"An Energy-Efficient Slotted Sense Multiple Access Broadcast Protocol for Reliable Command Delivery in Dynamic Wireless Sensor Networks,"* Sensors, vol. 19, no. 5, doi: 10.3390/s19051236.

[11]. J. Lipman, P. Boustead, and J. Chicharo, *"Reliable optimised flooding in ad hoc networks,"* in Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication (IEEE Cat. No.04EX710), vol. 2, pp. 521-524, (2004), doi: 10.1109/CASSET.2004.1321940.

[12]. O. Ugurlu, D. Tanir, and E. Nuri, *"A better heuristic for the minimum connected dominating set in ad hoc networks,"* in 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT), pp. 1-4, (2016), doi: 10.1109/icaict.2016.7991751.

[13]. B. Zeng, L. Yao, and Y. He, *"An Energy-efficient Broadcast Control Protocol for Wireless Sensor Networks,"* in 2009 IEEE International Conference on Networking, Architecture, and Storage, pp. 3-8, (2009), doi: 10.1109/NAS.2009.9.

[14]. N. Abramson, *"THE ALOHA SYSTEM: another alternative for computer communications,"* In Proceedings of fall joint computer conference (AFIPS '70 (Fall)). ACM, New York, NY, USA, 281-285, (1970).

[15]. S. Kim and D. Eom, *"Link-State-Estimation-Based Transmission Power Control in Wireless Body Area Networks,"* IEEE journal of biomedical and health informatics, vol. 18, (2013), doi: 10.1109/JBHI.2013.2282864.

[16]. F. Entezami, M. Tunicliffe, and C. Politis, *"Find the Weakest Link: Statistical Analysis on Wireless Sensor Network Link-Quality Metrics,"* IEEE Vehicular Technology Magazine, vol. 9, no. 3, pp. 28-38, (2014), doi: 10.1109/MVT.2014.2333693.

[17]. M. K. Maroti, B.; Simon, G.; Ledeczi, A, *"The flooding time synchronization protocol,"* Proc. ACM SenSys, Baltimore, pp. 39-49, (2004).

[18]. Datasheet for the CC2420 radio component. Available online: http://www.ti.com/product/CC2420.

TÓM TẮT

Giao thức truyền bản tin điều khiển hiệu quả cho mạng cảm biến động

Trong mạng cảm biến không dây (WSNs), nút chủ (sink) phải truyền bản tin điều khiển tới các nút cảm biến để điều khiển hoạt động của toàn mạng. Tuy nhiên, sự phân bố rộng khắp và nguồn tài nguyên giới hạn của các nút dẫn tới những khó khăn và thành thức cho quá trình truyền bản tin điều khiển. Bài báo này, chúng tôi đề xuất một giao thức truyền tin điều khiển hiệu quả về năng lượng dựa trên cấu trúc cây ở đó số lượng các nút cảm biến không có con là cực đại, và một khe thời gian chia sẻ được gắn theo số bước của nút cảm biến tới nút chủ. Hơn nữa, khe thời gian này còn được chia thành các khe thời gian nhỏ để các nút có cùng số bước cạnh tranh trước khi truyền tin, do đó, giảm sự xung đột. Khi một nút cảm biến di chuyển nó sẽ sử dụng khe thời gian chia sẻ ứng với vị trí mới để truyền nhận lệnh điều khiển. Các kết quả mô phỏng chứng minh rằng, giải pháp đưa ra không chỉ đạt độ tin cậy cao của bản tin cần truyền mà còn tiết kiệm năng lượng.

Từ khoá: Bản tin điều khiển; Nút cảm biến không có con; Sự di chuyển của nút; Khe thời gian chia sẻ; Năng lượng.