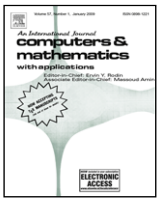
Computers and Mathematics with Applications 63 (2012) 48–59

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa



# Shortest hop multipath algorithm for wireless sensor networks

Onur Yilmaz [a], Sercan Demirci [a], Yagiz Kaymak [b,*], Serkan Ergun [a], Ahmet Yildirim [c,d]

[a] *International Computer Institute, Ege University, 35100 Bornova-Izmir, Turkey*
[b] *Faculty of Engineering and Computer Sciences, Izmir University of Economics, 35330 Balcova-Izmir, Turkey*
[c] *Department of Mathematics, Ege University, 35100 Bornova-Izmir, Turkey*
[d] *Department of Mathematics and Statistics, University of South Florida, Tampa, FL 33620-5700, USA*

ARTICLE INFO

*Article history:*
Received 24 August 2011
Received in revised form 25 October 2011
Accepted 25 October 2011

*Keywords:*
Shortest hop path
Braided multipath
Data dissemination
Routing
Distributed spanning tree algorithms
$\beta$ synchronizer

ABSTRACT

Shortest hop or distance path is one of the most common methods used for relaying messages in a wide variety of networks. It provides an efficient message relaying to destination in terms of energy and time. There are many algorithms for constructing shortest hop or distance path. However, according to our knowledge, no algorithm for constructing a shortest hop multipath for wireless sensor networks (WSNs) has yet been proposed in the literature. In this paper, we propose a novel distributed shortest hop multipath algorithm for WSNs in order to generate energy efficient paths for data dissemination or routing. The proposed algorithm generates shortest hop braided multipath to be used for fault-tolerance or load-balancing. It guarantees the BFS tree and generates near optimal paths in $O(V.D + V)$ message complexity and $O(D^2)$ time complexity regarding the communication costs towards the sink after termination of algorithm.

© 2011 Elsevier Ltd. All rights reserved.

## 1. Introduction

Wireless sensor networks (WSNs) have gained worldwide attention in recent years due to many technological advances and their military and commercial application potential [1]. These networks consist of individual nodes able to interact with their environment by sensing or controlling physical parameters [2]. These sensor nodes are small, with limited processing and computing resources, and they are inexpensive compared to traditional sensors. These sensor nodes can sense, measure, and gather information from the environment and, based on local decision processes, can transmit the sensed data to the user. Since the sensors operate on battery power, it is very important to use the energy of sensors efficiently to increase the network lifetime. Most of the energy of sensors is consumed by transmission of the data packets generated by that sensor or relaying the packets of the other sensors. Therefore, finding optimal transmission paths from each sensor to the destination is an essential task [3].

### 1.1. Motivation

Shortest hop or distance path is one of the most common methods used for relaying messages in all kinds of networks. It provides an efficient message relaying to destination in terms of energy and time. There are many algorithms for constructing shortest distance paths including the algorithms of Dijkstra and Bellman–Ford. The distributed Bellman–Ford (DBF) and the termination detection added version of DBF, Chandy–Misra [4], were proposed for distributed systems

[*] Corresponding author. Tel.: +90 232 488 82 61; fax: +90 232 488 84 75.
*E-mail addresses:* onur.yilmaz@ege.edu.tr (O. Yilmaz), sercan.demirci@ege.edu.tr (S. Demirci), yagiz.kaymak@ieu.edu.tr (Y. Kaymak), serkan.ergun@ege.edu.tr (S. Ergun), ahmet.yildirim@ege.edu.tr (A. Yildirim).

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.10.070

many years ago. It is obvious that the constructing shortest distance path has higher overhead than shortest hop path. In particular, DBF was used in [5–7] and Chandy–Misra in [8] to generate shortest distance paths in WSNs due to the distributed structure. However, their overheads are significantly high and the message complexity of these algorithms is unbounded since the algorithms are asynchronous. In [9], the algorithm of Chandy–Misra was synchronized and the message complexity is $O(V.E)$. Thus, these algorithms are not suitable for WSNs due to energy problem. On the other hand, shortest hop path algorithms are used instead of shortest distance path algorithms by many protocols for WSNs. In [10,11], the shortest hop paths are constructed in the PEQ and ICE protocol respectively. Even the method used for shortest hop path construction is straightforward and easy; since the algorithm is asynchronous, the message complexity cannot be calculated. Furthermore, due to lack of termination detection, the initiator node cannot know whether the algorithm terminates.

Multipath routing techniques have been discussed in the literature for several years. There are two different approaches to construct multipaths between two nodes. One is the classical node-disjoint multipath, where the alternate paths do not intersect each other. Another approach builds many braided paths and there are typically no completely disjoint paths but rather many partially disjoint alternate paths.

### *1.2. Contributions*

In this paper, we propose a novel distributed shortest hop multipath algorithm for WSNs. As far as we are aware, no algorithm for constructing a shortest hop multipath for WSNs has yet been proposed. The proposed algorithm generates shortest hop braided multipath in order to be used for fault-tolerance or load-balancing. In order to obtain the BFS tree, the Chang–Roberts distributed spanning tree (CRDST) algorithm is synchronized. While the synchronized algorithm is being performed, the braided multipaths are constructed. Thus, every constructed path reaches the initiator node by the shortest hop.

We use the $\beta$ synchronizers in order to synchronize the asynchrony CRDST algorithm. $\beta$ synchronizer is optimal in terms of message complexity but it is inefficient in terms of time complexity. Since sending a message consumes a great deal of energy, and energy efficiency is a major challenge in WSNs, we use the $\beta$ synchronizer instead of other synchronization techniques. In [12], the $\beta$ synchronizer is described as a layer based synchronization method. There is only a single synchronizer node responsible for producing the *pulse* messages. The execution proceeds layer by layer. At the beginning of a turn, all nodes in a layer begin their operations. When they finish, they send a message to inform the synchronizer node, which then sends a new *pulse* message to pass the turn to the next layer. The algorithm has also termination detection. Therefore, the initiator node can know whether the algorithm terminates.

## **2. Related works**

In [11], routing protocols are categorized including data attribute based, flat, geographical, hierarchical, multipath and QoS-based. Well known routing protocols in WSNs are SPIN [13], Directed Diffusion [14], LEACH [15], and GAF [11]. SPIN and Directed Diffusion are data-centric protocols. SPIN labels the data with meta-data and this information is exchanged among the neighbors. Directed diffusion is a query based protocol which uses interest messages. Sink sends interest messages to the network and if the data of node matches with the interest message, it sends gradients towards the sink. While the source is sending gradients, the path is formed during this process. LEACH is a hierarchical protocol which uses clustering. In LEACH, the clusters and cluster heads of sensor networks are formed based on the receiving signal strength. Data is routed to sink through the cluster heads. GAF is a location based protocol where location information is used for energy efficiency. If the region to be sensed is known, the query is only diffused at that region. Thus, energy dissipation is prevented.

Recent works include [16,11]. [16] is a data-centric routing protocol which is also fault tolerant. [11] is a hierarchical protocol which is an event-driven, query based protocol.

Shortest hop or distance path is one of the most commonly used methods for relaying messages in a wide variety of networks. It provides an efficient message relaying to destination in terms of energy and time. The DBF and Chandy–Misra [4] were proposed many years ago for distributed systems. DBF is used in [5–7] and Chandy–Misra in [8] for generating shortest distance paths in WSNs. Shortest hop path algorithms are used instead of shortest distance path algorithms by many protocols for WSNs. In [10,11], the shortest hop paths are constructed in the PEQ and ICE protocol respectively.

Multipath routing techniques have been discussed in the literature for several years. There are two different approaches including disjoint and braided multipaths. Generally, the multipath is used in [17–20] for fault-tolerance, load-balancing and security.

Very recently, in soliton theory, the linear superposition principle applies to Hirota bilinear equations to construct soliton solutions [21]. As known, Hirota bilinear equations are linked to binary Bell polynomials. Ma and Fan [21] showed that a linear superposition principle of exponential waves applied to Hirota bilinear equations, under some additional condition on the exponential waves and possibly on the polynomial as well. This is an interesting mathematical problem in the study of polynomials whose zeros form a vector space, and it determines one class of Hirota bilinear equations possessing the linear superposition principle of exponential waves. They analyzed some such specific Hirota bilinear equations.

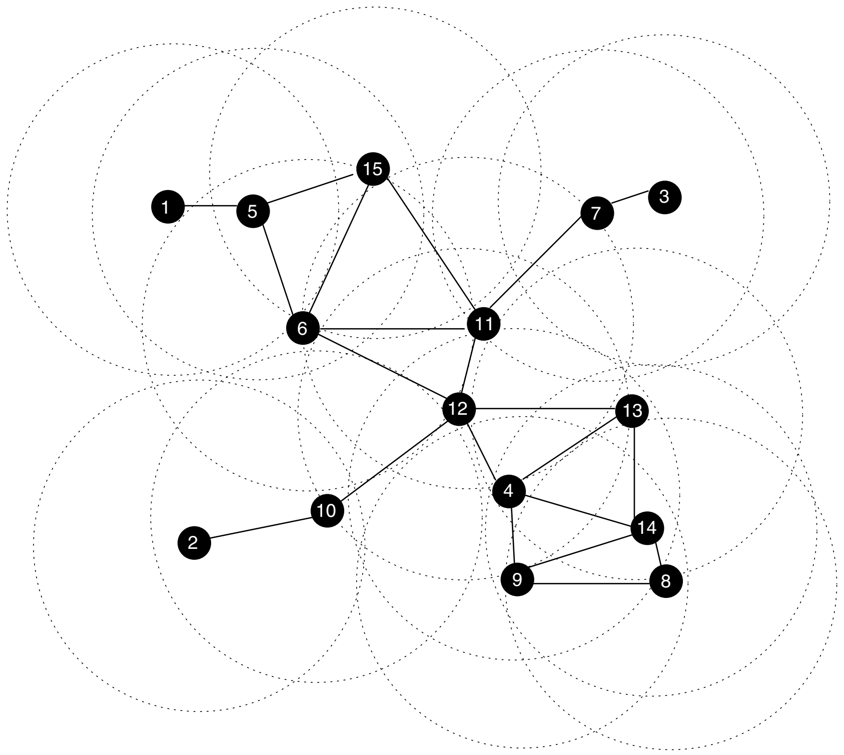


**Fig. 1.** Unit disk graph model.

## 3. Distributed shortest hop multipath algorithm (SHM)

### 3.1. Network modeling

The following assumptions are made about the network.

- Each node has distinct *node_id*.
- The nodes are stationary.
- The transmission range of a node is modeled with a circle.
- Links between nodes are symmetric. Thus if there is a link from $u$ to $v$, there exists a reverse link from $v$ to $u$.
- Nodes do not know their positions. They are not equipped with a position tracker like a GPS receiver.
- All nodes are equal in terms of processing capabilities, radio, battery and memory.

Based on these assumptions, wireless ad hoc and sensor networks are generally modeled as Unit Disk Graphs (UDG) [22]. In a UDG as depicted in Fig. 1 [23], there is an edge $\{u, v\}$ if the maximum transmission radius of both $u$ and $v$ is at least $|uv|$, their Euclidean distance. Each node in network can adjust its transmission power to any value between zero and its maximum transmission power level. Only undirected (symmetric) edges are considered in order to send acknowledgment over same link.

### 3.2. Overview

The Chang–Roberts distributed spanning tree (CRDST) algorithm is a starting point for our Shortest Hop Multipath (SHM) algorithm. The CRDST algorithm works asynchronously and does not guarantee Breadth-First-Search (BFS) tree formation after termination. It uses two types of messages, *probe* and *echo*. The *probe* message is used by all nodes to make membership request. The *echo* message is a reply of a received *probe* message and it has an acknowledgment or negative acknowledgment meaning. The algorithm is terminated by the initiator in case of receiving all *echo* messages from neighbors. When the CRDST algorithm terminates, it generally constructs an unbalanced spanning tree.

Our proposed SHM algorithm is a synchronized version of a CRDST algorithm and works synchronously. Synchronously means that this algorithm runs step by step. In other words, sink node does not start the next step until it receives all expected messages. The SHM algorithm generates a BFS tree after each execution. To guarantee the BFS tree formation, we use the $\beta$ synchronizer approach [12]. Figs. 2 and 3 display a sample execution of the SHM algorithm.

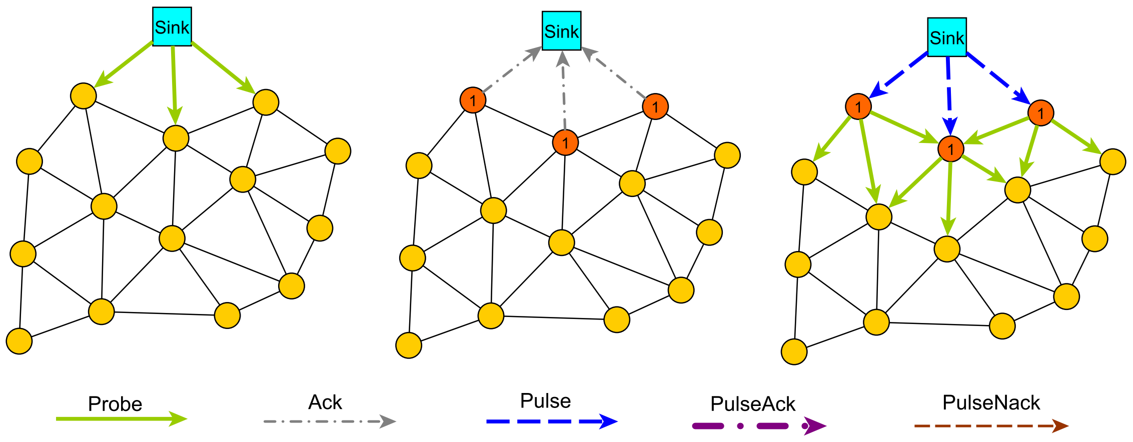


**Fig. 2.** (a) *probe* message (start of the SHM) (b) *ack* message (c) *pulse* message (sink gives the turn to first layer).

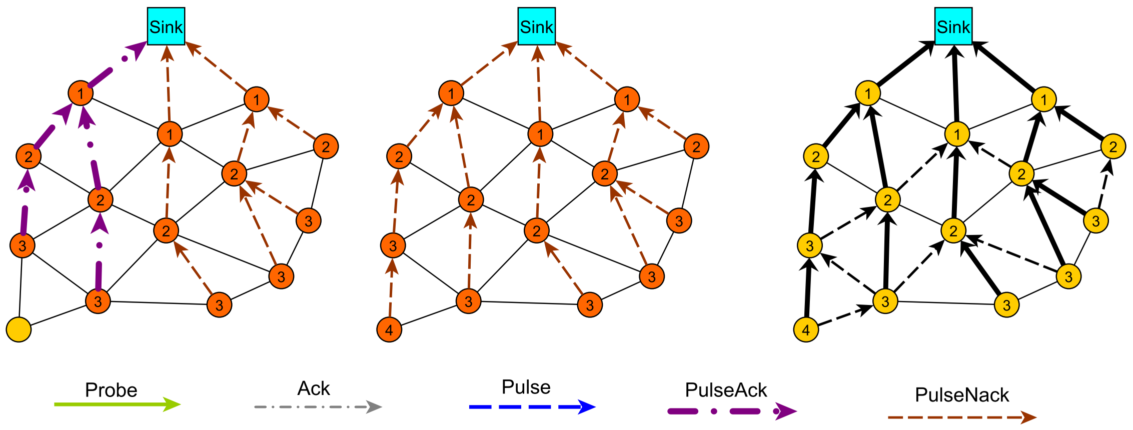


**Fig. 3.** (a) *pulseAck* and *pulseNack* messages (b) *pulseNack* message (termination) (c) Generated shortest hop multipath.

### 3.3. *Description of the algorithm*

In this section, we introduce the SHM algorithm and how it works. There are two kinds of nodes in this algorithm, initiator (sink), acting as the root, and non-initiator acting as an internal or a leaf node in the spanning tree. Initiator node is responsible for initializing and finalizing of spanning tree construction. Also there are five types of messages that are used in the algorithm. These are *probe*, *ack*, *pulse*, *pulseAck* and *pulseNack*. *probe* message is used in order to notify the level value of the receiver and solicit to be the parent of the receiver. *ack* message is a reply of a *probe* message. If a node accepts another node as a parent, it sends back an *ack* message. *pulse* message is generated by initiator node and it uses this message in order to inform which layer has the turn of sending *probe* message. The last two messages, *pulseAck* and *pulseNack*, are generally replies to a received *pulse* message, and are in fact used to determine the termination detection. *probe*, *ack* and *pulse* messages are transmitted as broadcasting. However, only *ack* message includes the ID of receiver node because overhearing technique is used for decreasing message complexity. On the other hand, *pulseAck* and *pulseNack* messages are transmitted as unicast. In the SHM algorithm, we have the following assumption, *initiator*(*sink*) node discards all the *probe* and *pulse* messages.

The pseudocode of the algorithm for non-initiator and initiator node are shown in Algorithms 1 and 2 respectively. We also use FSM diagram to show that the SHM algorithm is deadlock-free and works properly. Nodes which are in topology have two roles: initiator and non-initiator. Fig. 4 displays the finite state machine of initiator and non-initiator nodes.

## 4. Analysis

**Theorem 1.** *The message and time complexities of the SHM algorithm are* $O(V.D+V)$ *and* $O(D^2)$ *respectively where D is diameter of the network from the sink to furthest leaf and V is the node count of the network.*

**Algorithm 1** Non-initiator Node

```
upon node $u$ receives probe message from node $v$
  if node $u$ has not set its parent yet then
    set node $v$ as parent
    set level as received level value in probe message
  else if level of node $u$ == received level value in probe message then
    add node $v$ as alternate parent
  end if
end upon
upon node $u$ receives pulse message from node $v$
  if node $v$ is the parent of node $u$ then
    if node $u$ has no neighbor then unicast pulseNack to parent
    else if level of node $u$ == received level value in probe message then
      broadcast probe by the level value in message = level of node + 1
    else if level of node $u$ $<$ received level value in probe message then
      broadcast the received pulse message for forwarding to children nodes
    end if
  end if
end upon
upon node $u$ receives ack message from node $v$
  if node $v$ does not send ack message to node $u$ then
    remove node $v$ from neighbors
  end if
  if sum of received ack == neighbor number then unicast pulseAck to parent
end upon
upon node $u$ receives pulseAck message from node $v$
  if sum of received pulseAck and pulseNack == children number then
    unicast pulseAck to parent
  end if
end upon
upon node $u$ receives pulseNack message from node $v$
  if sum of received pulseAck and pulseNack == children number then
    if node $v$ has not received any pulseAck message then
      unicast pulseNack to parent
    else
      unicast pulseAck to parent
    end if
  end if
end upon
```

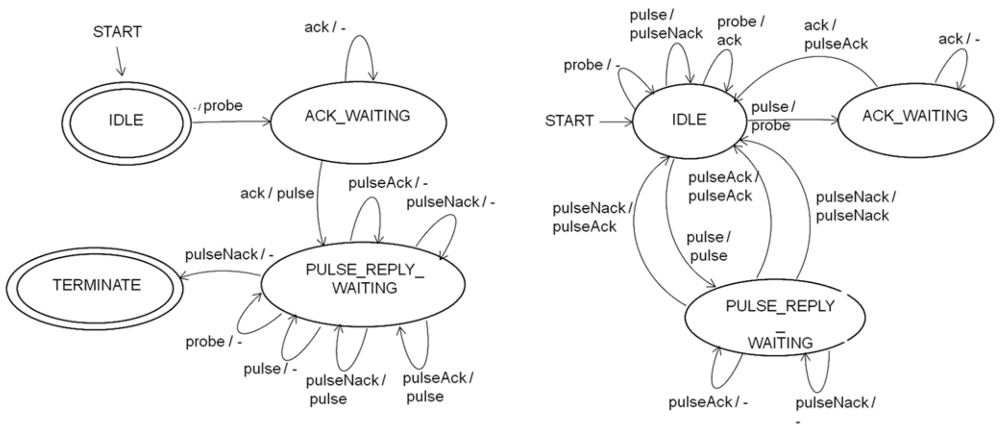


**Fig. 4.** (a) Initiator (sink) node (b) Non-initiator node.

**Proof.** For $p \geq 0$, let $V_p$ denote the set of vertices in layer $p$ of the tree $T$ constructed by the algorithm. *Probe* messages of phase $p$ are sent only on $V_p$ and the vertices of $T_p$ are traversed twice for broadcast (*Pulse*) and convergecast messages

**Algorithm 2** Initiator (sink) Node

```
1: broadcast *probe* message
2: **upon** sink receives **ack** message
3:   **if** sum of received ack messages == neighbor number **then**
4:     broadcast *pulse* message in order to give the turn to first level
5:   **end if**
6: **end upon**
7: **upon** sink receives ***pulseAck*** message
8:   **if** sum of received *pulseAck* and *pulseNack* == neighbor number **then**
9:     broadcast *pulse* message in order to give the turn to next level
10:  **end if**
11: **end upon**
12: **upon** sink receives ***pulseNack*** message
13:   **if** sum of received *pulseAck* and *pulseNack* == neighbor number **then**
14:     **if** all received messages are *pulseNack* **then**
15:       Terminate
16:     **else**
17:       broadcast *pulse* message in order to give the turn to next level
18:     **end if**
19:   **end if**
20: **end upon**
```

(*PulseAck* and *PulseNack*). In sum, we get;

$$Message(Phase\ p) := O(V + V_p).$$

For all the $1 \le p \le D$;

$$\begin{aligned} Message(SHM) &:= \sum_1^p Message(Phase\ p) \\ &:= \sum_1^p O(V + V_p) \\ &:= O(V.D + V). \end{aligned}$$

Clearly, broadcast (*pulse*) and convergecast (*pulseAck* and *PulseNack*) stages each take $p$ time units and the *probe* and *ack* message broadcasting stage takes additional two time units.

$$\begin{aligned} Time(Phase\ p) &:= 2p + 2 \\ Time(SHM) &:= \sum_1^p Time(Phase\ p) \\ &:= \sum_1^p 2p + 2 \\ &:= O(D^2). \qquad \square \end{aligned}$$

In this paper, we use random networks to find the average hop of a constructed BFS tree after the SHM algorithm terminates. Random networks have been studied for several decades. Paul Erdös and Alfréd Rényi introduced what are known as "classical" random networks, or *Erdös–Rényi* networks. The basic idea is that we consider a simple, connected graph on $n$ vertices, and that every two vertices are adjacent with some probability $p$ [24]. *Erdös–Rényi* networks are used to simulate the SHM algorithm.

It can be seen in Fig. 5 that as the edge probability approaches one, the randomly constructed graph converges to a fully connected graph. In a fully connected graph, every node has adjacency with every other node. In this case, it is expected that the SHM algorithm terminates in one hop. However, as the edge probability decreases, node degrees of the graph reduces. This results in an increase on the depth of the BFS tree produced by the SHM algorithm. Figs. 6–9 show that the depth of constructed BFS tree is logarithmically increasing with node count. They also reveal that increase in $p$ converges the expected hop count of BFS tree to one. Thus the expected hop count of the BFS tree constructed by the SHM algorithm can be asymptotically bounded by $O(\log(n)/(np))$. We do not present here the mathematical proof of this bound for the sake of conciseness; the reader could find this proof in [25].

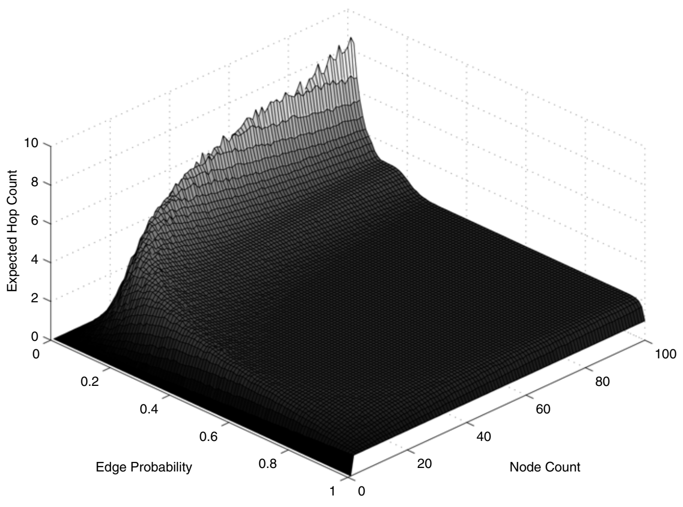


**Fig. 5.** Expected hop count of the SHM algorithm.

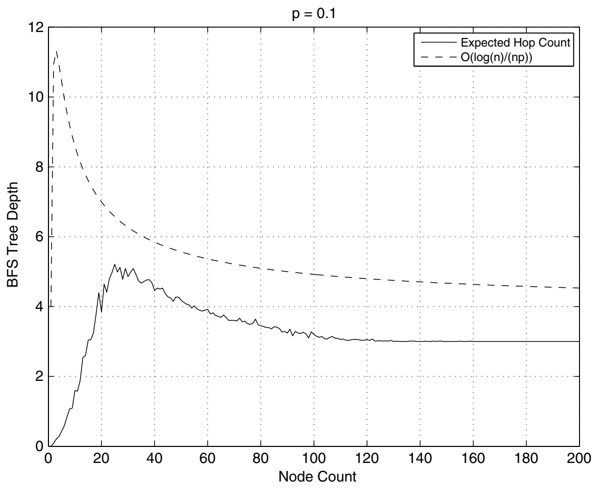


**Fig. 6.** Expected BFS tree depth when $p = 0.1$.

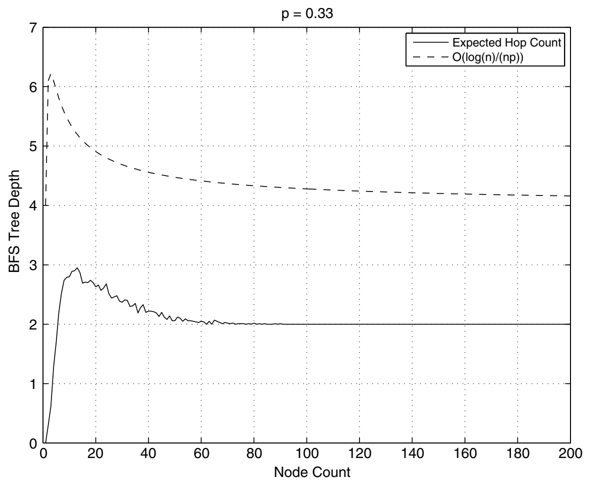


**Fig. 7.** Expected BFS tree depth when $p = 0.33$.

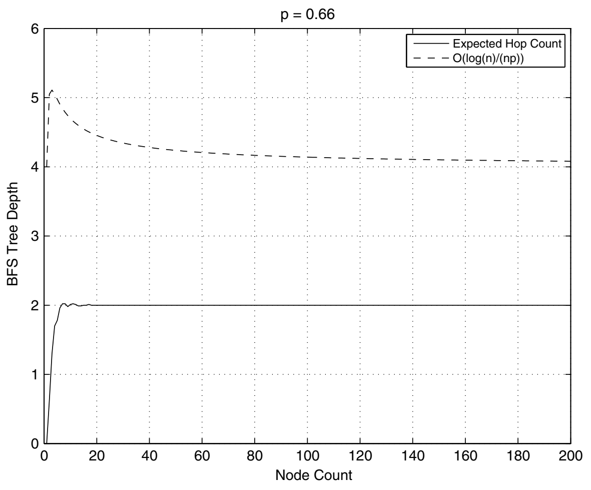


**Fig. 8.** Expected BFS tree depth when $p = 0.66$.

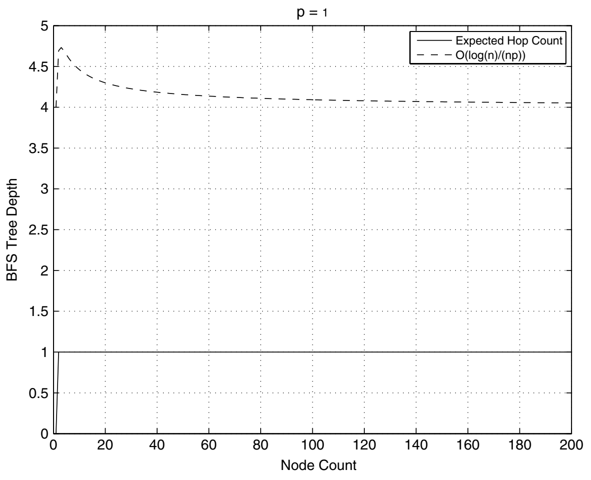


**Fig. 9.** Expected BFS tree depth when $p = 1.0$.

**Table 1**
Experimental study parameters.

| Area | Random |
|---|---|
| Sink position | (0, 0) |
| Number of sensors | 100–300 |
| MAC | Perfect MAC |
| Transmission power | 0.395 W |
| Receiving power | 0.360 W |
| Initial energy | 10 J |
| Transmission range | 75 m |

## 5. Simulations

### 5.1. Simulation details

In order to evaluate our algorithm, to measure its performance and to find the overheads, we used ns-2 network simulator. The metrics which we decided on were running time and energy dissipation of algorithm, average hop counts, average path lengths, average delays of generated paths and network lifetime. We executed the algorithm for each metric in 100, 150, 200, 250 and 300 nodes.

Network area is $1000 \times 1000$ m. Randomly generated topologies were used. The sink was located at (0,0) origin. The initial energy of nodes were assigned to 10 J. Perfect MAC was used in the MAC layer. Table 1 displays the simulation parameters.

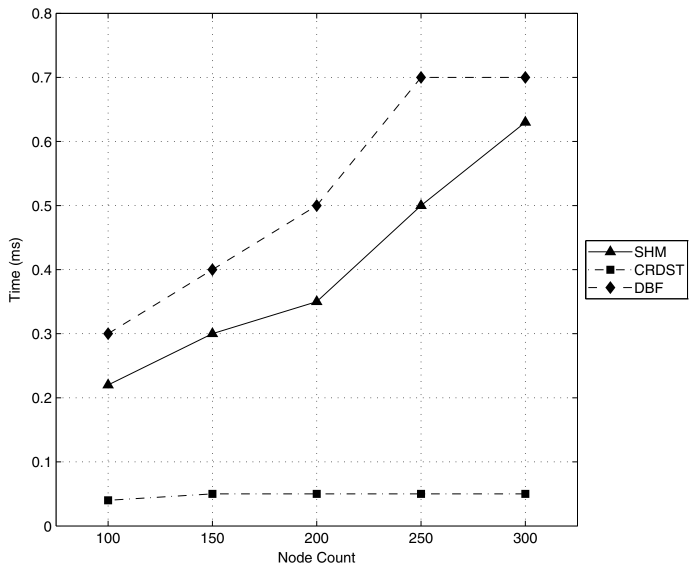


**Fig. 10.** Average running times of SHM, CRDST and DBF algorithms.

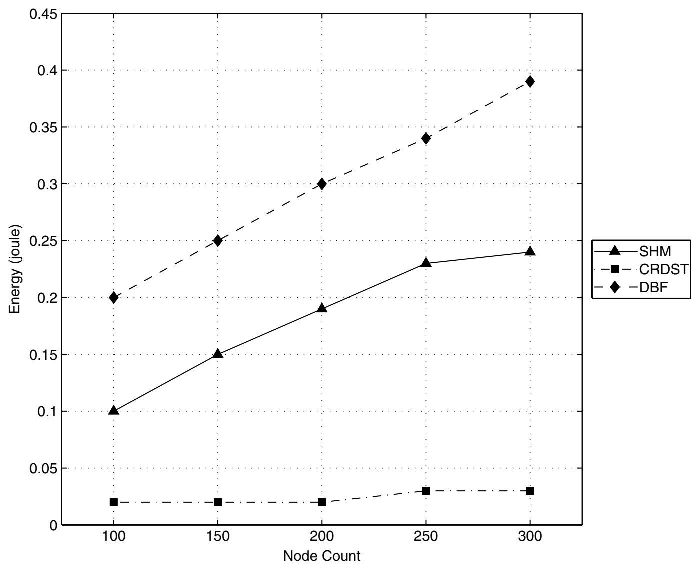


**Fig. 11.** Average energy dissipations of SHM, CRDST and DBF algorithms.

## *5.2. Simulation results*

The running time and energy dissipation metrics are related to execution of SHM, DBF and CRDST algorithms. Hop count, path length, delay and network lifetime metrics are related to the properties of generated paths by these algorithms.

Fig. 10 displays the running time results of the SHM, DBF and CRDST algorithms for different node counts ranging from 100 to 300 nodes. Fig. 10 shows us that SHM and DBF algorithms take longer time than CRDST algorithm. In addition, the DBF algorithm takes longer time than the SHM algorithm to finish its operation. The running time of the CRDST algorithm is nearly 0.1 s at all node counts but, on the other hand the running time of SHM and DBF algorithms increase linearly ranging from 0.2 to 0.6 s and 0.3 to 0.7 s, respectively. It is clear, therefore, that the SHM algorithm takes less time than the DBF algorithm.

Fig. 11 displays the energy dissipation results of the SHM, DBF and CRDST algorithms for different node counts ranging from 100 to 300 nodes. Fig. 11 shows that SHM and DBF algorithms consume more energy than the CRDST algorithm. The energy dissipation of the CRDST algorithm is nearly 0.1 J at all densities but, on the other hand the energy dissipation of SHM and DBF algorithms increase from 0.1 to 0.24 J and 0.2 to 0.4 J respectively for each density. It is clear that the SHM algorithm consumes less energy than the DBF algorithm.

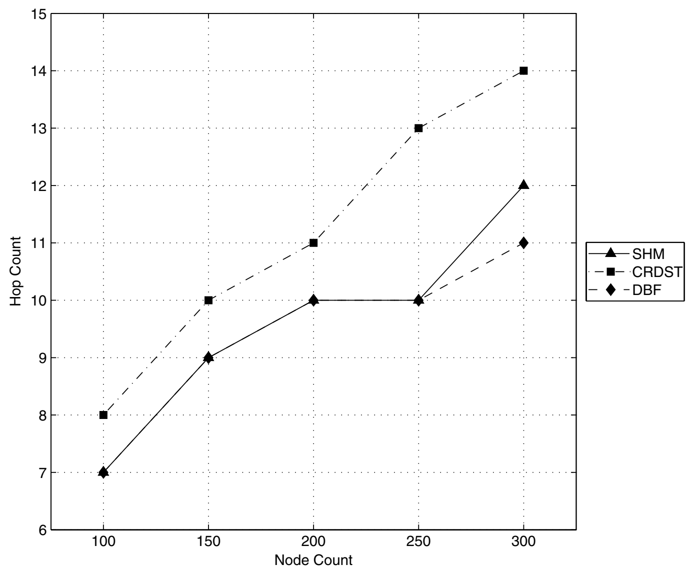


**Fig. 12.** Average hop counts of generated paths by SHM, CRDST and DBF algorithms.

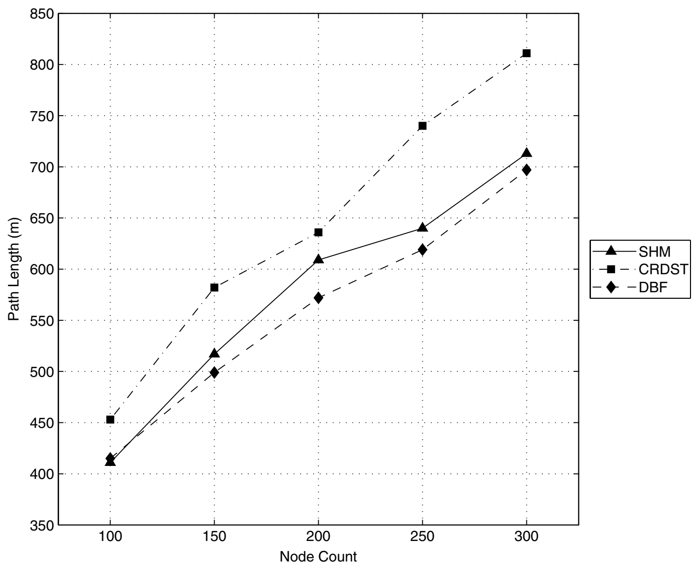


**Fig. 13.** Average lengths of generated paths by SHM, CRDST and DBF algorithms.

The two metrics above are initial values for execution of SHM, DBF and CRDST algorithms. The two figures above indicate that in order to construct a BFS tree, more time than the running time of CRDST is needed and more energy than the energy dissipation of CRDST is dissipated, but since the generated paths through the spanning tree have near optimal paths regarding lengths, low energy is dissipated and low delay is obtained while routing data towards sink. It is obvious that the DBF algorithm dissipates more energy than the SHM algorithm.

Fig. 12 displays the average hop counts of the generated paths from SHM, DBF and CRDST algorithms for different node counts ranging from 100 to 300 nodes. Hop count is an important metric for energy efficiency because sending and receiving messages entails high cost. Thus, if a path has optimal hop count, it consumes less energy and this also affects the network life time. Since the SHM algorithm guarantees the BFS tree, the SHM algorithm generates optimal hop count paths. Fig. 12 shows that the hop count of paths of SHM and DBF algorithms are nearly same.

Fig. 13 displays the average path lengths of the generated paths from SHM, DBF and CRDST algorithms for different node counts ranging from 100 to 300 nodes. Another important metric for energy efficiency is path length because the energy consumption is dependent on the distance the data has to travel in wireless communication. Thus, in order to save energy during data transmission, distances must be as short as possible. Since the BFS tree is constructed by the SHM algorithm, minimum hop is guaranteed towards sink. In addition, minimum length is also guaranteed among the layers. Fig. 13 shows

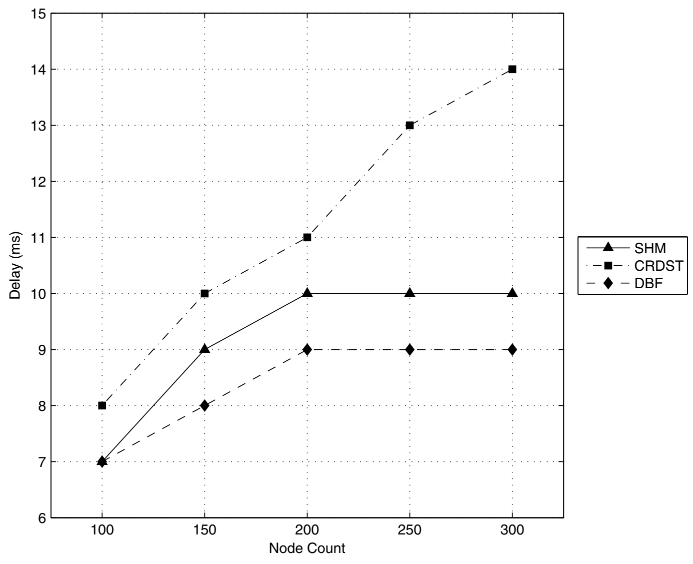


**Fig. 14.** Average delay of generated paths by SHM, CRDST and DBF algorithms.

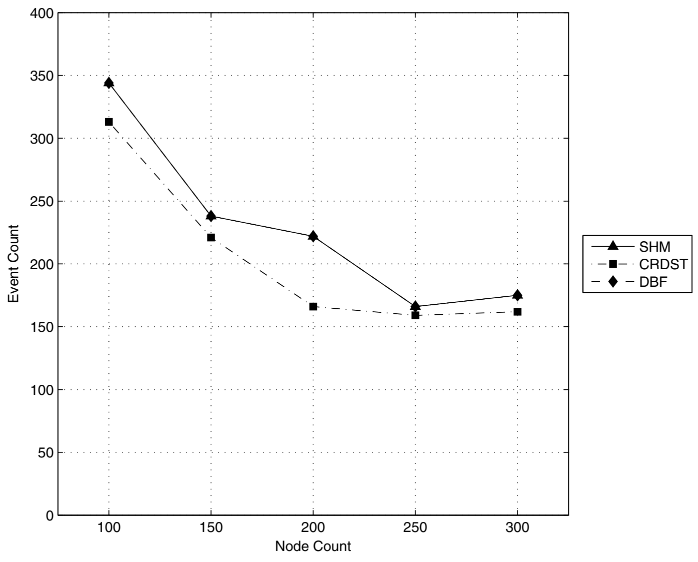


**Fig. 15.** Network lifetime of generated paths by SHM, CRDST and DBF algorithms.

us that SHM generates shorter paths than the CRDST algorithm but, since the DBF generates shortest distance paths, its paths are shorter but the difference is reasonable.

Fig. 14 displays the average delay of the generated paths from SHM, DBF and CRDST algorithms for different densities ranging from 100 to 300 nodes. Fig. 15 displays the average network lifetime (the time it takes for the first node to die) of the generated paths from SHM, DBF and CRDST algorithms for different densities ranging from 100 to 300 nodes. Since the time taken is affected by various metrics, we use the event numbers in order to compare the lifetimes of networks.

Fig. 15 shows that SHM and DBF algorithms increase the network lifetime because they generate more optimal paths compared to the CRDST algorithm in terms of path length and hop count. The network lifetime of SHM and DBF are almost identical. Network lifetime of SHM, DBF and CRDST algorithms decreases while network density increases because the length and delay of generated paths increase as intensity increases. Due to this, the delivered message count decreases as the node count decreases.

## 6. Conclusion

In this paper, we propose a novel distributed shortest hop multipath algorithm for WSNs. As far as we are aware, no one has yet proposed an algorithm which construct a shortest hop multipath for WSNs. The proposed algorithm generates shortest hop braided multipath for the purpose of fault-tolerance or load balance. Another contribution of this paper is that the algorithm includes termination detection. None of the currently proposed shortest hop algorithms for WSNs feature termination detection and therefore the sink cannot determine when/whether the entire network has finished determining the paths. If sink wants to disseminate data to network, it has to wait an unknown time since it does not know whether the paths have been constructed. In addition to contributions, the SHM algorithm has a bounded message and time complexity compared to widely used shortest hop path algorithms in the literature.

The simulation results show that, although compared to the CRDST algorithm, more energy and time are consumed by SHM and DBF algorithms, the paths they generate consume less energy and have less delay than the paths of the CRDST algorithm. The SHM algorithm spends less time and consumes less energy than DBF because the message and time complexity of DBF are very high. In addition, the SHM algorithm generates almost identical paths to DBF algorithm.

## References

[1] C. Li, L. He, Implementation and emulation of distributed clustering protocols for wireless sensor networks, in: Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing, IWCMC'07, 2007, pp. 266–271.

[2] H. Karl, A. Willig, Protocols and Architectures for Wireless Sensor Networks, John Wiley & Sons, 2005.

[3] M. Kalantari, M. Shayman, Energy efficient routing in wireless sensor networks, 2004.

[4] K.M. Chandy, J. Misra, Distributed computation on graphs: shortest path algorithms, Commun. ACM 25 (1982) 833–837.

[5] J.H. Chang, L. Tassiulas, Maximum lifetime routing in wireless sensor networks, IEEE/ACM Trans. Netw. 12 (2004) 609–619.

[6] J.H. Chang, L. Tassiulas, Energy conserving routing in wireless ad-hoc networks, in: INFOCOM, 2000, pp. 22–31.

[7] S. Coleri, P. Varaiya, Fault tolerant and energy efficient routing for sensor networks, in: GlobeCom, 2004.

[8] O. Yilmaz, K. Erciyes, Distributed weighted node shortest path routing for wireless sensor networks, in: Recent Trends in Wireless and Mobile Networks, Communications in Computer and Information Science, 2010, 2010, pp. 304–314.

[9] K.B. Lakshmanan, K. Thulasiraman, M.A. Comeau, An efficient distributed protocol for finding shortest paths in networks with negative weights, IEEE Trans. Softw. Eng. 15 (1989) 639–644.

[10] A. Boukerche, R. Werner Nelem Pazzi, R. Borges Araujo, Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments, J. Parallel Distrib. Comput. 66 (2006) 586–599.

[11] A. Boukerche, Algorithms and Protocols for Wireless Sensor Networks, Wiley-IEEE Press, 2008.

[12] D. Peleg, Distributed Computing: A Locality-Sensitive Approach, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[13] W.R. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, in: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, MobiCom'99, 1999, pp. 174–185.

[14] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, IEEE/ACM Trans. Netw. 11 (2003) 2–16.

[15] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, An Application-Specific protocol architecture for wireless microsensor networks, IEEE Trans. Wireless Commun. 1 (4) (2003) 660–670.

[16] I. Chatzigiannakis, A. Kinalis, S. Nikoletseas, Fault-tolerant and efficient data propagation in wireless sensor networks using local, additional network information, J. Parallel Distrib. Comput. 67 (2007) 456–473.

[17] D. Ganesan, R. Govindan, S. Shenker, D. Estrin, Highly-resilient, energy-efficient multipath routing in wireless sensor networks, in: Proceedings of the 2nd ACM International Symposium on Mobile ad Hoc Networking & Computing, MobiHoc'01, 2001, pp. 251–254.

[18] M.K. Marina, S.R. Das, Ad hoc on-demand multipath distance vector routing, Wireless Commun. Mobile Comput. 6 (7) (2006) 969–988.

[19] Y. Liu, W.K.G. Seah, A scalable priority-based multi-path routing protocol for wireless sensor networks, Int. J. Wirel. Inf. Netw. 12 (1) (2005) 23–33.

[20] J. Ben Othman, L. Mokdad, Enhancing data security in ad hoc networks based on multipath routing, J. Parallel Distrib. Comput. 70 (2010) 309–316.

[21] W.X. Ma, E. Fan, Linear superposition principle applying to Hirota bilinear equations, Comput. Math. Appl. 61 (2011) 950–959.

[22] B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs, Discrete Math. 86 (1990) 165–177.

[23] O. Yilmaz, O. Dagdeviren, K. Erciyes, Interference-aware dynamic algorithms for energy-efficient topology control in wireless ad hoc and sensor networks, Comput. J. 54 (2011) 1398–1411.

[24] P. Erdös, A. Rényi, On random graphs, Publ. Math. 6 (1959) 290–297.

[25] F. Chung, L. Lu, The diameter of sparse random graphs, Adv. Appl. Math. 26 (2001) 257–279.