Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2009, Article ID 697373, 14 pages
doi:10.1155/2009/697373

Research Article

Distributed and Location-Based Multicast Routing Algorithms for Wireless Sensor Networks

Hakki Bagci and Ibrahim Korpeoglu

Department of Computer Engineering, Bilkent University, Ankara, Turkey

Correspondence should be addressed to Ibrahim Korpeoglu, korpe@cs.bilkent.edu.tr

Received 30 July 2009; Accepted 19 October 2009

Recommended by Athanasios T. Vasilakos

Multicast routing protocols in wireless sensor networks are required for sending the same message to multiple different destinations. In this paper, we propose two different distributed algorithms for multicast routing in wireless sensor networks which make use of location information of sensor nodes. Our first algorithm groups the destination nodes according to their angular positions and forwards the multicast message toward each group in order to reduce the number of total branches in multicast tree which also reduces the number of messages transmitted. Our second algorithm calculates an Euclidean minimum spanning tree at the source node by using the positions of the destination nodes. The multicast message is forwarded to destination nodes according to the calculated MST. This helps in reducing the total energy consumed for delivering the message to all destinations by decreasing the number of total transmissions. Evaluation results show that the algorithms we propose are scalable and energy efficient, so they are good candidates to be used for multicasting in wireless sensor networks.

1. Introduction

Nowadays, low-cost tiny sensor nodes which communicate over wireless channels have become available due to advances in hardware technology. The idea of collaboration of these tiny sensors has enabled wireless sensor networks (WSNs). Sensor networks are composed of large number of densely deployed sensor nodes. Usually WSNs are self organizing and do not require a fixed infrastructure, so communication in a sensor network is accomplished in an ad hoc manner. For example, sensor nodes can be randomly deployed from an airplane to inaccessible terrains. With cooperation of sensor nodes, data from the monitored region can be gathered to a base station [1]. These key features of sensor networks make them a promising technology which will be used in many areas and will become indispensable in our daily lives in near future. Wireless sensor networks have many application areas such as military applications, environment monitoring, health monitoring, and also commercial applications for home and industry [2].

Routing protocols for wireless sensor networks target basically two main problems: Data dissemination and data gathering. Data dissemination includes the process of routing the queries or data into the sensor network, while data gathering consists of collecting data sensed by the nodes and delivering it to a base station. There are many protocols proposed for data dissemination and data gathering problems in wireless sensor networks [3–8].

In this paper, we study a specific type of data dissemination problem: multicasting. Given a set of destination nodes, we try to deliver a message from sink node to all these destination nodes. We are focusing on location-based multicast solutions. We offer two algorithms for location-based multicast routing in wireless sensor networks. We think that location-based routing is useful, since when monitoring a region usually we want to know where the data is sensed. In addition, for some applications we may be interested in some subregions of the whole region, so we may want to send our queries to these subregions.

The algorithms we propose in this paper are Location-Based Multicasting with Direction (LBM-D) and Location-Based Multicasting according to Minimum Spanning Tree (LBM-MST). As it is common for most location-based routing algorithms, we assume that all nodes in the network know their own locations and their neighbors' locations. Basically, LBM-D groups the destinations according to the angles they make with the current node and it selects a next node for each group to forward the multicast message. Next node selection algorithm is based on local greedy decisions to make progress toward the destination nodes. LBM-MST calculates an Euclidean Minimum Spanning Tree that covers all destination nodes and uses the LBM-D algorithm to follow the paths in the constructed MST. MST is calculated once in the sink node and distributed to the network with multicast messages.

We also evaluate our algorithms and compare them with a similar location-based multicasting algorithm named Position Based Multicasting (PBM) [9]. We choose simulation as our evaluation methodology and compare the three algorithms on the same scenarios in terms of number of total transmissions, average end-to-end delay, traffic overhead and success rates. Simulation results show that our proposed algorithms perform well and can be used as location-based multicasting protocols for wireless sensor networks.

The remainder of this paper is organized as follows. In the next section, we describe some of the work related to location-based routing and multicasting. In Section 3, we first give a detailed description of the PBM protocol [9] that we used for comparisons. We describe how we have implemented it. Then in the same section, we describe our proposed algorithms in detail. In Section 4, we report the evaluation results of the algorithms, and we conclude the paper in Section 5.

2. Related Work

In this section we will briefly describe some related work on location-based multicasting for wireless ad hoc and sensor networks.

Multicasting in a network can be defined as sending a message to multiple destinations which are probably located in different regions of the network. The difference between multicasting and ordinary unicast routing is that in multicasting we send the same message (data) to more than one destination. Most of the time the main challenge in multicasting is delivering the message to all destinations with minimum total number of hops, because it is directly related with the total power consumption, which is an essential concern especially for wireless sensor networks.

There is a similar problem called geocasting which is indeed a special type of multicasting where destinations are located within a certain region of the network. In this paper we assume destinations can be at any point in the network, not clustered in a certain region.

Zhang, Jia, Huang and Yang proposed four different heuristic schemes, namely, single branch regional flooding, single branch multicast tree, cone-based forwarding area multicast tree, and MST-based single branch multicast tree for location-based multicasting in sensor networks [10]. In single branch regional flooding (SARF), the nearest node to the center of the multicast region is defined as access point (AP) and messages are first routed to the AP by using Dijsktra's shortest path algorithm. Then AP floods the message within the multicast region. This approach can only work for geocasting problem obviously, and is not distributed because Dijsktra's shortest path algorithm requires global knowledge of the network topology. In addition it is not power efficient since it uses flooding within the multicast region. The second approach defined in [10] is single branch multicast tree (SAM) in which an AP is selected as the node which is closest to the sink. The multicast message is forwarded to the AP as in the same way used in SARF. For broadcasting the message in multicast region, SAM constructs a multicast tree whose root node is AP. While constructing the multicast tree, in each step, the node with the maximum number of effective neighbors is selected among the neighbors of the multicast tree in the multicast region. Effective neighbors are the neighbors which reside in multicast region and are not included in the multicast tree yet. In cone-based forwarding area multicast tree (CoFAM) method, a cone-based forwarding area is defined from the sink to the given region. Only nodes in the forwarding area forward the multicast messages, nodes outside the forwarding region do not relay any packets. As in SAM, a multicast tree is formed, but in this case the tree is rooted at the sink node which covers the forwarding area. In MST-based single branch multicast tree (MSAM) approach, a minimum spanning tree is constructed in multicasting region and multicast packages are routed to the root of the MST by Dijsktra's shortest path algorithm. The approaches proposed by Zhang et al. are all designed for geocasting and require knowledge of the global network topology, which makes them quite impractical for sensor networks.

Another position based multicast algorithm (PBM) was proposed by Mauve [9]. This algorithm tries to build a multicast tree by applying a greedy neighbor selection approach. Each relay node receiving the multicast message evaluates a cost function for each subset of its neighbors to decide on the best subset to forward the multicast message. Thus the algorithm has an exponential computational cost to evaluate the 2^n subsets of a node, where n is the number of neighbors. We compare our algorithms with PBM, because the problem it solves is exactly the same with ours and it is one of the best localized geographic multicast routing algorithms to date.

SPBM [11], which aims to design a scalable position based multicasting protocol, mainly focuses on managing multicast groups in a scalable way. However, SPBM uses separate unicast geographic routing for each destination and interchanges the routing tables between neighbors, which decrease the efficiency and scalability with increasing number of multicast groups.

DSM [12] is another location-based multicast routing protocol for mobile ad hoc networks proposed by Basagni, Chlamtac and Syrotiuk. It is a source-routing based scheme in which multicast tree is constructed at source node and calculated tree is sent with multicast packets in a decoded way. Each node receiving this message decodes the multicast tree that comes with the message and routes the message according to this tree. The weak point of this approach is that each node should know the location information of all other nodes in the network. This information is required to construct the entire multicast tree. In order to maintain this information each node holds a GPS cache that stores the updated location information of all other nodes.

Sanchez, Ruiz, Liu, and Stojmenovic proposed GMR [13], geographic multicast routing protocol for wireless sensor networks. GMR's neighbor selection scheme depends on the cost over progress ratio where cost is defined as the number of neighbor nodes selected for relaying. The progress is the overall reduction of the remaining distances to destinations. Neighbor selection algorithm is based on a greedy set merging scheme. However this algorithm requires testing d^3 subsets of neighbors of a node in the worst case, where d is the number of destinations.

Wu and Candan proposed a geographical multicast routing protocol (GMP) for wireless sensor networks in which routing is done according to virtual Euclidean Steiner trees rooted at the transmitting nodes [14]. Each transmitting node locally computes a virtual Euclidean Steiner tree using a reduction ratio heuristic. According to this tree and local knowledge regarding to neighbors of the current node, destinations are divided into groups. For each group, a next node is selected and a multicast message is forwarded to that group via the selected node. In this approach, some points of virtual Euclidean Steiner tree does not correspond to the actual sensor nodes, so some extra work is performed to deal with this virtual destinations. In addition, although GMP uses an efficient reduction ratio heuristic to compute virtual Euclidean Steiner trees, this calculation is performed by all transmitting nodes which makes GMP quite inefficient in terms of power consumed by processing at sensor nodes.

3. Location-Based Multicasting Algorithms

In this section we propose two different algorithms for location-based multicast routing in wireless sensor networks. We also provide the details of another algorithm that is proposed by Mauve et al. [9] and which is originally developed for mobile ad hoc networks. We apply this algorithm to wireless sensor networks with slight differences and use it for comparison with our algorithms.

3.1. Preliminaries. Before describing the algorithms in detail, we want to introduce some concepts used in our implementations and some assumptions that we have made. We assume that every node in the network has the information of its own location in terms of geographical coordinates, and the position of each destination is known to the sender, as usual for position based routing algorithms. In addition, we assume that each node has the information of its neighbors' locations or can detect them on the fly with a simple handshake protocol.

In order to keep track of the transmitted messages, we need to define a *MessageSignature* which uniquely identifies each multicast message. A *MessageSignature* is composed in the following way for a message that needs to be delivered to n destinations. Suppose a message has a *MessageID* which is generated uniquely when the message is created at the node which starts multicasting, and we have a string representation of the location information (like coordinates), *RepLoc*. Then the corresponding message signature is:

$$\begin{aligned} &MessageID + RepLoc_{dest_1} + RepLoc_{dest_2} \\ &+ \cdots + RepLoc_{dest_n}. \end{aligned} \tag{1}$$

In each node we hold a list of sent *MessageSignatures* (*SMS*) to keep track of the messages that are sent/forwarded by the node. *SMS* list can be cleared after a multicasting session is ended. In practice a node should clear its *SMS* list when it receives a message whose *MessageSignature* starts with a *MessageID* that is different than the last *MessageSignature*'s *MessageID*.

In our implementations we use a data structure called *Dictionary* which holds $\langle key, value\rangle$ pairs. We call such pairs as *entries*. Given a key k, we can access the corresponding value and we can sort entries according to the keys. The entries hold in *Dictionary* can be enumerated so we can iterate on the dictionary entries. *Dictionary* data structure is based on a hash table where the search time is asymptotically constant when all the keys are unique.

3.2. Position Based Multicasting (PBM). In this section, we describe Position Based Multicasting (PBM) algorithm proposed by Mauve et al. [9] and how we have implemented it in order to apply it to wireless sensor networks. PBM is a distributed algorithm which makes local decisions to find next nodes to forward the multicast packets. In order to find the next nodes, the following formula is evaluated in each node which receives a multicast message:

$$f(w) = \lambda\frac{|w|}{|N|} + (1-\lambda)\frac{\sum_{z\in Z}\min_{m\in w}(d(m,z))}{\sum_{z\in Z}(d(k,z))}, \tag{2}$$

where k is the forwarding node, N is the set of all neighbors of k, W is the set of all subsets of N, w is a subset of N, Z is the set of all destinations and $d(x,y)$ is a function which calculates the distance between nodes x and y.

First part of (2) gives the normalized number of next hop nodes and the second part calculates the total remaining distance to all destinations normalized to the distance from current node to all destinations. Parameter $\lambda \in [0,1]$ is used to combine these criteria linearly. When λ is close to 0, multicast messages are split earlier, while λ is close to 1, messages will be split as late as possible.

When a node receives a message it evaluates this function for each subset of its neighbors w and find a subset w' which minimizes the function. Although (2) gives the subset w', the neighbors which will take the message next, it does not say which destination will be assigned to which neighbor in subset w'. In order to solve this problem we use a heuristic described later in this section.

Outline of our implementation of PBM is given in Algorithm 1. This pseudo-code is run on each node which receives a multicast message M. After reception of the message M, it is checked to see if its *MessageSignature* resides in Sent Message Signatures (SMS) cache. If it exists in SMS cache, this means that this message was sent by

```
(1)  for all received message $M$ do
(2)    if $M_{signature}$ exists in Sent Message Signatures
       cache then
(3)      Drop the message
(4)      Continue for the next message
(5)    end if
(6)    if $M_{dests}$ contains $this$ then
(7)      Remove $this$ from $M_{dests}$
(8)      if $M_{dests}$ is empty then
(9)        Continue for the next message
(10)     end if
(11)   end if
(12)   $Next\_Nodes \leftarrow$ FindNextNodes($M_{dests}$)
(13)   $Asgns \leftarrow$ AssignDestNodes($M_{dests}$,$Next\_Nodes$)
(14)   for all assignment $ASGN$ in $Asgns$ do
(15)     $newMessage \leftarrow$ Create a message for $ASGN$
(16)     Add $M_{signature}$ into Sent Message Signatures
         cache
(17)     Forward $newMessage$ to $ASGN_{nextNode}$
(18)   end for
(19) end for
```

Algorithm 1: Greedy Multicast Forwarding in PBM.

```
(1) $P \leftarrow$ Create all subsets of $N$
(2) $minCost \leftarrow infinity$
(3) $optSubset \leftarrow \{\}$
(4) for all subset $S$ of $P$ do
(5)   $cost \leftarrow$ CalculateCost($S$, $destinations$)
(6)   if $minCost$ is greater than $cost$ then
(7)     $minCost \leftarrow cost$
(8)     $optSubset \leftarrow S$
(9)   end if
(10) end for
(11) return $optSubset$
```

Algorithm 2: Finding Next Nodes for Destinations in PBM.

```
(1)  Create an empty dictionary $Asgns$
(2)  Create a list $uncoveredDests$
(3)  Copy $destinations$ to $uncoveredDests$
(4)  for all node $n$ in $nextNodes$ do
(5)    $nearestDestNode \leftarrow$ Find nearest destination
       node to $n$
(6)    Add a new entry $e$ to $Asgns$ such that
       $e_{key}$ is $n$ and $e_{value}$ is $\{nearestDestNode\}$
(7)    Remove $nearestDestNode$ from $uncoveredDests$
(8)  end for
(9)  for all node $d$ in $uncoveredDests$ do
(10)   $nearestNextNode \leftarrow$ Find nearest next node to $d$
(11)   Add $d$ into the list $Asgns[nearestNextNode]$
(12)   Remove $d$ from $uncoveredDests$
(13) end for
(14) return $Asgns$
```

Algorithm 3: Assignment of Destination Nodes to Next Nodes in PBM.

this node before so we should not resend it, because PBM would send it in the same way as the former one which will create an infinite loop. Therefore we drop this message and start waiting for the next incoming multicast message. If *MessageSignature* of the message does not exist in *SMS* cache, we start to process the message. First we check if the node that received the message is in the destination list, in other words if it is one of the destinations of the multicast message. If so, current node is removed from the list of destinations, and after this operation if no destination remains, it means that all destinations have received the multicast message on this branch of the multicast tree. In this case we start to wait for another incoming multicast message.

After the checks mentioned above, if the message has destinations to be forwarded, next nodes among the neighbors of the current node are selected by *FindNextNodes* procedure. Having found the next nodes, destinations are assigned to next nodes by the procedure *AssignDestNodes*. These assignments are hold in a dictionary data structure. An assignment is composed of two parts, first one is the node that the message will be forwarded and the second part consists of destinations that are assigned to the node. For each assignment a message is created and forwarded to the next node of the corresponding assignment.

Pseudo-code for the procedure *FindNextNodes* that finds the nodes to forward the message next is given in Algorithm 2. It takes the list of destinations as input to be used in cost calculations. First, all subsets of the neighbors of the current node are found. Then for each subset S, a cost is calculated by *CalculateCost* function and the subset with the minimum cost is found and returned. *CalculateCost* function calculates the cost of sending a multicast message to a subset of neighbors of the current node according to the cost function given in (2).

Having found the subset with minimum cost, destinations are assigned to the nodes in the subset according to the procedure *AssignDestNodes* in Algorithm 3. The method used by PBM is not explicitly mentioned in [9], so we designed our own method for assigning the destinations to the next nodes. The procedure *AssignDestNodes* takes the list of destinations and next nodes found by the procedure *FindNextNodes* as input and produces a list of assignments as output. Given the destinations and the next nodes, we make sure that number of destinations are equal or greater than the number of next nodes. In other words our method must assign at least one destination to each next node. To guarantee this condition, we first make a pass over the next nodes and assign each of them a destination. According to our method, we select the closest destination to a next node and assign this destination to it at first pass. After the first pass, if some unassigned destinations remain, we make another pass over the unassigned destinations and select the closest next node for each destination. In this way every next node has at least one destination and no destination remains unassigned. A next node with its destination(s) constitutes an assignment and these assignments are added to a list and resulting assignments list is returned.

```
(1)  for all received message $M$ do
(2)      if $M_{signature}$ exists in Sent Message Signatures
         cache then
(3)          Drop message $M$
(4)          Continue for the next message
(5)      end if
(6)      if $M_{dests}$ contains $this$ then
(7)          Remove $this$ from $M_{dests}$
(8)      end if
(9)      if $M_{destinations}$ is not empty then
(10)         $node\_group\_list$ ← GroupDestinations($M_{dests}$)
(11)         for all entry $e$ in $node\_group\_list$ do
(12)             $nextNode$ ← SelectNextNeighbor($e$)
(13)             $newMessage$ ← Create a message for $e$
(14)             Add $M_{signature}$ into SMS cache
(15)             Forward $newMessage$ to $nextNode$
(16)         end for
(17)     end if
(18) end for
```

Algorithm 4: Location-Based Multicasting with Direction.

```
(1)  $SortedDestinations$ ← Sort $M_{destinations}$ according to
     angles they make with $this$
(2)  $current\_angle$ ← 0
(3)  $node\_group$ ← Create an empty node group
(4)  $node\_group\_list$ ← Create an empty node group list
(5)  for all entry $e$ in $SortedDestinations$ do
(6)      if $node\_group$ is empty then
(7)          $current\_angle$ ← $e_{angle}$
(8)          Add $e_{node}$ to $node\_group$
(9)      else
(10)         if $current\_angle + \alpha$ is greater than $e_{angle}$ then
(11)             Add $e_{node}$ to $node\_group$
(12)         else
(13)             $node\_group_{angle}$ ← $current\_angle$
(14)             Add $node\_group$ to $node\_group\_list$
(15)             $node\_group$ ← Create a new empty
                 node group
(16)             $current\_angle$ ← $e_{angle}$
(17)             Add $e_{node}$ to $node\_group$
(18)         end if
(19)     end if
(20) end for
(21) if $node\_group$ is not empty then
(22)     Add $node\_group$ to $node\_group\_list$
(23) end if
(24) return $node\_group\_list$
```

Algorithm 5: Grouping Destinations in LBM-D.

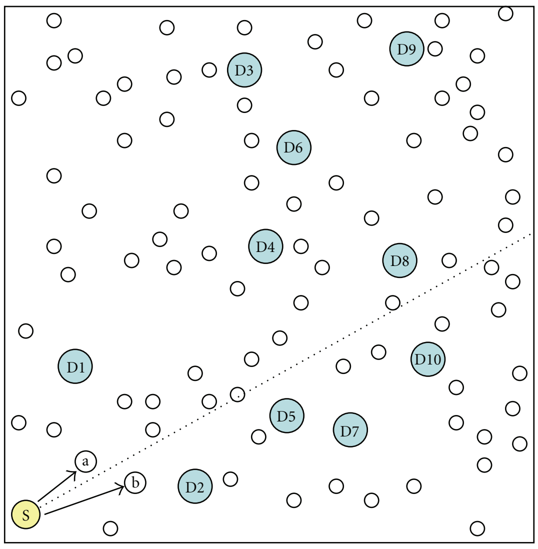
3.3. Our Proposed Algorithm 1: Location-Based Multicasting with Direction (LBM-D). In this section, we describe first of our location-based multicasting algorithms called Location-Based Multicasting with Direction (LBM-D). As its name implies, we use the direction information of the destinations to forward the multicast messages. Like PBM, LBM-D is also a distributed algorithm which makes local greedy decisions to make progress toward destination nodes. Basically it groups the destinations according to their directions and for each group it creates a multicast message and forwards it to corresponding next nodes. This algorithm mainly consists of two parts. First part generates the groups of destinations and second part finds the next nodes for each group of destinations. LBM-D is summarized in Algorithm 4 and sub-procedures used by LBM-D is given in Algorithms 5 and 6. We first explain the main flow of the algorithm shown in Algorithm 4 and then get into details of other procedures.

In LBM-D, whenever a multicast message M is received, we first check to see if its *MessageSignature* exists in SMS cache. If *MessageSignature* resides in SMS cache, message will not be forwarded further and progress will stop in the current branch with some unreached destinations.

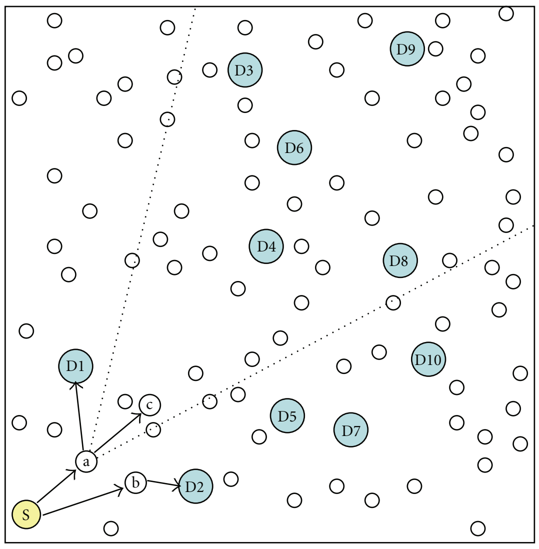
If *MessageSignature* does not exist in SMS cache, we check if the current node is in the list of destinations of the multicast message. If the current node resides in the destinations list, it means that multicast message is delivered to one of the destinations. In this case, we remove the current node from the destinations list. After all, if the destinations list is not empty, remaining destinations are grouped by the procedure *GroupDestinations*, which takes list of destinations as input and produces a dictionary whose entries are pairs of angle value and a group of destinations in the form of $\langle angle, listOfDestinations\rangle$. Having grouped the destinations into the dictionary *node_group_list*, a next node is selected for each group of destinations by the procedure *SelectNextNeighbor*. Then, a separate multicast message is created for each group and sent to the corresponding next node. In addition, *MessageSignature*s of the sent messages are written into SMS cache of the current node.

Our approach for grouping the destinations according to the angles they make with the current node is implemented as seen in Algorithm 5. Every node that receives a multicast message with some remaining destinations runs this procedure. Output of this procedure is a dictionary structure that holds groups of destinations for corresponding angle values. The angles depend on the given parameter *alpha* which determines the angle ranges used for partitioning the destinations. For example, if *alpha* is 45 degrees, this means that maximum 8 (360/45) groups can be constituted from the destinations. In this case, destinations that make an angle with the current node in the range of [0–45) will be grouped together, and that are in the range of [45–90) will be grouped together, and so on. Hence, parameter *alpha* affects the branching behavior of the algorithm in terms of number of branches taken by a forwarding node.

The procedure *GroupDestinations* given with Algorithm 5 takes the list of destinations as input and sorts the destinations according to the angles they make with the current node. *SortedDestinations*, which is a dictionary structure, holds the sorted destinations where the key of each entry is the angle and the value part is the corresponding destination node. Output of this procedure is a list of groups of destinations. The *node_group* variable in Algorithm 5 holds the destinations that are in the same group and *node_group_list* variable holds *node_group*s constructed by the procedure. The **for** loop in the procedure



(a)



(b)

Figure 1: A sample run of LBM-D with 10 destinations.

```
(1)  $minNode \leftarrow null$
(2)  $minDiff \leftarrow infinity$
(3)  $currDiff \leftarrow infinity$
(4)  for all neighbor $n$ in $this_{neighbors}$ do
(5)      if $destinations$ contains $n$ then
(6)          return $n$
(7)      end if
(8)      $currAngle \leftarrow$ Calculate angle between $this$ and $n$
(9)      $currDiff \leftarrow |angle - currAngle|$
(10)     if $minDiff$ is greater than $currDiff$ then
(11)       if $n$ has more than 1 neighbor then
(12)         $minDiff \leftarrow currDiff$
(13)         $minNode \leftarrow n$
(14)       end if
(15)     end if
(16) end for
(17) return $minNode$
```

Algorithm 6: Selecting Next Neighbor in LBM-D.

traverses all the entries in *SortedDestinations* and if the angle of the destination lies in the range (*current_angle*, *current_angle* + *alpha*), it adds the destination to the current *node_group*. The *current_angle* variable is only changed when the current *node_group* is empty and the *node_group* becomes empty when the angle of the current destination does not lie in the range given above. In this case, we add the current *node_group* to the *node_group_list* and start constructing a new *node_group* whose angle is set by the current destination's angle. Having created a new *node_group*, this destination is added to it and the loop continues with the next destination in *SortedDestinations*. After traversing all entries *SortedDestinations*, the last constructed *node_group* remains not included, so we add it into *node_group_list* just after the **for** loop terminates.

After partitioning the destinations, we should determine the next node for each group of destinations to forward the multicast message. Our neighbor selection algorithm is presented in Algorithm 6. The procedure *SelectNextNeighbor* takes the list of destinations, which is a group constructed by *GroupDestinations*, and the angle associated with this group as input and returns a node to forward the message next. *SelectNextNeighbor* includes a **for** loop that iterates over all neighbors of the current node which is depicted as *this* in Algorithm 6. At the beginning of the **for** loop it is checked whether destination list contains the current neighbor. If so, **for** loop is terminated and this neighbor is returned. Otherwise, the angle between *this* and current neighbor is calculated and the neighbor node which makes the closest angle to the given *angle* is selected. In order to make a progress, the selected node must have more than one neighbor including the neighbor from where the message is received, otherwise it would not find a neighbor to forward to message further. To guarantee this, we also make a check while selecting the next neighbor node. If such a node cannot be found, *SelectNextNeighbor* procedure will return *NULL* and the progress at this branch will stop.

A sample run of LBM-D algorithm is given in Figures 1 and 2. A sample topology with a source node and 10 destinations (D1,...,D10) is shown in Figure 1(a). The dashed lines originating from the source node (depicted as S) divide the region into subregions according to the *alpha* which is used to group the destinations. Solid arrows depict the transmission of multicast messages from one node to another. According to the case in Figure 1(a), destinations D1, D3, D4, D6, D8, D9 and D2, D5, D7, D10 reside in the same subregion. Therefore source node S makes two groups of destinations, (D1, D3, D4, D6, D8, D9) and (D2, D5, D7, D10). Figure 1(a) also shows the branching according to the groups that are created by the source node S. A multicast message with target destinations (D1, D3, D4, D6, D8, D9) is sent to node *a* and another multicast message is sent to node

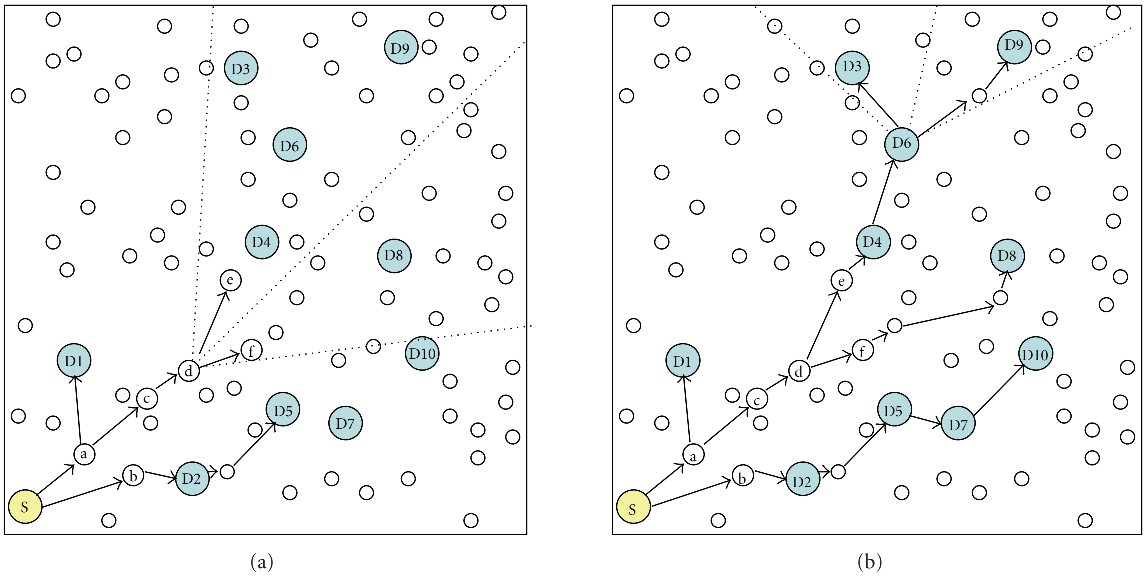


FIGURE 2: A sample run of LBM-D with 10 destinations cont'd.

b with target destinations (D2, D5, D7, D10). In Figure 1(b), node *a* decides to branch further with destination D1 and remaining destinations (D3, D4, D6, D8, D9) on the current branch. Destination D1 is reached and a multicast message with destinations (D3, D4, D6, D8, D9) is forwarded to the neighbor node *c* which is selected according to the neighbor selection algorithm described in Algorithm 6. On the other branch, node *b* forwards the message to D2 and destination D2 is reached. In Figure 2(a), node *d* takes another branch with two groups of destinations (D3, D4, D6, D8, D9) and (D8). A multicast message with destinations (D3, D4, D6, D8, D9) is sent to node *e* and another multicast message with target destination D8 is sent to node *f*. Again nodes *e* and *f* are selected according to our neighbor selection scheme. On the other branch, D5 is reached and removed from the destination list of the corresponding multicast message. Figure 2(b) shows the final routes taken by the LBM-D algorithm in order to deliver the multicast message to all destinations.

The weak point of the PBM algorithm was its neighbor selection strategy which requires 2^n comparisons, where n is the number of neighbors of a node. This approach makes PBM unscalable, since in dense networks the number of comparisons will increase so rapidly, which will require excessive amount of energy. However, in sensor networks power is a scarce resource which should be used very carefully. With LBM-D, we propose a neighbor selection algorithm whose running time is linear on the number of neighbors in the worst case. This feature of LBM-D makes it scalable and a better algorithm to be used for location-based multicasting in wireless sensor networks.

3.4. Our Proposed Algorithm 2: Location-Based Multicasting with Direction According to MST (LBM-MST). Algorithms described in previous sections use only local information and depending on this information make greedy decisions to deliver the multicast messages to destinations. LBM-D introduced a new scalable neighbor selection approach which makes it more scalable than PBM. LBM-MST is another algorithm that we propose which is based on LBM-D. LBM-MST is also a distributed algorithm, but it also uses the location information of the destinations in a global way and routes the multicast messages according to a minimum spanning tree which is calculated at the source node. Before starting to route packets, locations of all destinations are known to the source node. LBM-MST makes use of this information by calculating an Euclidean minimum spanning tree of all destinations. The information about the constructed MST is also forwarded with multicast messages, so that MST is only calculated once at the originator. The additional space overhead (per destination) posed by MST in a packet is just a pointer to another destination, which determines the parent-child relationship between destinations. Furthermore, at branching points, destinations that a message contains are divided into pieces, so the size of multicast messages do not notably increase for LBM-MST. The analysis of the message overhead caused by extra pointers is given in Section 4.1.4.

Basically, LBM-MST draws a routing path for LBM-D by pointing to the destination which the multicast message should be delivered next. The path between two destinations is found by LBM-D. In other words, LBM-MST acts as a driver for LBM-D. Outline of LBM-MST is given in Algorithm 7.

A multicast message in LBM-MST has both next destinations and all destinations. Next destinations are the immediate child nodes of the previous destination in the minimum spanning tree. All destinations are the nodes of the tree which is rooted at the previous destination according to MST. In Figure 3, for a multicast message that is sent from node S to D1, destination D1 is the next destination where all destinations are D1, D2, and D3.

```
(1)  for all received message $M$ do
(2)    if $M_{signature}$ exists in SMS cache then
(3)      if there exists a neighbor $n$ with an empty SMS
         cache then
(4)        Forward $M$ to $n$
(5)      else
(6)        Drop message $M$
(7)        Continue for the next message
(8)      end if
(9)    end if
(10)   if $M_{nextDests}$ contains $this$ then
(11)     Remove $this$ from $M_{nextDests}$
(12)   end if
(13)   if $M_{allDests}$ contains $this$ then
(14)     Remove $this$ from $M_{allDests}$
(15)     $M_{nextDests}$ ← GetImmediateChildren($M_{allDests}$)
(16)   end if
(17)   $node\_group\_list$ ← GroupDestinations($M_{nextDests}$)
(18)   for all entry $e$ in $node\_group\_list$ do
(19)     $nextNode$ ← SelectNextNeighbor($e$)
(20)     $newMessage$ ← Create a message for $e$
(21)     Add $M_{signature}$ into SMS cache
(22)     Forward $newMessage$ to $nextNode$
(23)   end for
(24) end for
```

Algorithm 7: Location-Based Multicasting with Direction According to MST.

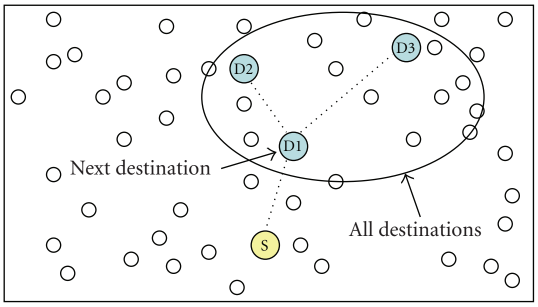


Figure 3: Next Destination and All Destinations in LBM-MST.

Before starting routing a multicast message, the source (e.g. base station) calculates the Euclidean MST using the locations of all destinations by using Prim's algorithm [15]. This information is hold in $M_{allDests}$. Then next destinations $M_{nextDests}$ are calculated by the procedure *GetImmediateChildren*. The first message is forwarded to next destinations with LBM-D. $M_{nextDests}$ are checked to see if the current node is one of the next destinations. If so it is removed from the list of next destinations and also from all destinations. After these operations, next destinations are calculated by *GetImmediateChildren* procedure for the current node. Having found the next destinations, they are grouped by the procedure *GroupDestinations* and multicast message for each group is forwarded to next neighbors as in the same way with LBM-D. The only difference with LBM-D is that in LBM-MST we group only the next destinations instead of all destinations. By this way we try to follow the paths drawn by the overall Euclidean MST constructed at the originator node.

Next node	User data	Dest_1	Parent_1	Dest_2	Parent_2	. . .	Dest_n	Parent_n

Figure 4: Multicast message used by LBM-MST algorithm.

```
(1) $childrenList$ ← Create an empty node list
(2) for all node $destination$ in $allDestinations$ do
(3)   if $destination_{Parent}$ = $this$ then
(4)     Add $destination$ to $childrenList$
(5)   end if
(6) end for
(7) return $childrenList$
```

Algorithm 8: Finding Immediate Child Nodes in LBM-MST.

The procedure *GetImmediateChildren* is a simple routine which traverses the list of all destinations and finds the immediate child nodes of the current node. As we mentioned before, multicast messages carry the MST information in $M_{allDests}$. This is achieved by holding a reference to the parent node of each destination in $M_{allDests}$ as shown in Figure 4. Fields indicated as *Parent_i* are pointers for the parent nodes of the corresponding destination nodes depicted as *Dest_n*. So if a destination's parent is the current node, we add it to the list of immediate children *childrenList*. This list is returned by the procedure when it is done.

A sample run of the LBM-MST Algorithm 8 is given in Figure 5. The topology of the network, which is the same with in Figure 1, is shown in Figure 5(a). The dashed lines in Figure 5(a) shows the Euclidean minimum spanning tree of 10 destinations. The solid arrows in Figure 5(b) depict the actual paths followed by the LBM-MST algorithm. As seen from Figure 5(b), LBM-MST tries to follow the Euclidean MST to minimize the number of total transmissions. In this sample case, LBM-D makes 19 transmissions to deliver the multicast message to all destinations, whereas LBM-MST requires only 15.

By following a minimum spanning tree of the destinations, we expect that LBM-MST will require less transmissions than LBM-D and PBM in order to deliver a message to all destinations. Since transmission is the most power consuming operation for sensor nodes, we expect LBM-MST will use less power than LBM-D and PBM. Therefore, sensor nodes in LBM-MST will die later which prolongs the network lifetime. However, we should expect a greater average end-to-end delay for LBM-MST, because messages are enforced to follow the branches of a tree instead of following a direct path. But for most of the sensor network applications, network lifetime is the prominent concern and end-to-end delay has a little importance compared to energy conservation.

4. Evaluation

In this section we report the results of the experiments done to evaluate our algorithms. We evaluate our algorithms by comparing them with PBM [9] algorithm and also with each other. In order to achieve this, we implemented a basic wireless sensor network simulator with C# language on .NET environment. It has also a graphical user interface which enables defining input parameters easily for single sample runs.

We evaluated our algorithms by comparing them with a similar location-based multicast routing algorithm, PBM [9]. For all scenarios, each algorithm is run with same set of parameters. For an iteration of an experiment, all three algorithms are run on the same network topology and with same destinations. We repeated the experiments many times, hence we did many iterations. At each iteration the random seed is changed to obtain unbiased results.

We mainly used the following metrics to evaluate our algorithms.

(i) *Success Rate*. We calculate success rate as the ratio of number of destinations that received the message divided by the total number of reachable destinations. In order to calculate a more reliable success ratio, it is necessary to omit the unreachable destinations due to topology of the network. Success rate is one of the most important metrics which shows how well the algorithms perform.

(ii) *Average End-to-End Delay*. The timespan that begins with the transmission of a message from sink and ends with the reception of the message by a destination is called end-to-end delay. Average end-to-end delay is the average of all end-to-end delays that occur for each destination. We do not calculate end-to-end delay in time units, instead we measure the end-to-end delay by the number of hops taken from the sink to a destination. We use this metric only to compare the algorithms. We do not calculate absolute end-to-end delays, since it is dependent to the hardware used by the sensors. We also neglect the time spent for processing in the sensors.

(iii) *Number of Total Transmissions*. Assuming each message hopping takes one transmission, we count the total number of transmissions made in order to the deliver the message from sink to all destinations. This metric is an indicator for the total power consumption due to transmissions and hence important for sensor network routing algorithms.

(iv) *Traffic Overhead*. We also measure the total bytes transmitted during all transmissions which plays an important role on total power consumption.

For each scenario we randomly deployed a set of sensor nodes on a 300×300 m area with a base station at the center. We take the transmission range of the sensor nodes as 35 m. By changing the number of sensors deployed and the number of destinations we created many different test cases. For each scenario, we repeated the experiments many times to obtain reliable results. The destinations are randomly chosen from the deployed sensors in each scenario.

Without loss of generality, in our experiments we assume that we know the locations of the destination nodes. In all of our scenarios, we are given a set of destination nodes whose locations are known and we try to deliver the message to all destinations, which is transmitted by the base station located in the center of the network.

We used three kinds of simulation scenarios in our experiments.

(1) All destination nodes are located in the same region. In this scenario, all destinations are in the same subregion of the network. In other words, all destinations are located close to each other, thus constituting a destination region. We choose a fixed subregion to locate all destinations for each run to be able to see the difference of the algorithms clearer while changing the network topology in each time.

(2) Destinations are located in separated regions. In this scenario, destinations are partitioned into groups and these groups are located in separate subregions of the network. In our simulations we tried to locate the destinations almost at the same locations to see the difference of the algorithms for each run, while changing the topology each time.

(3) Destinations are located randomly. In this scenario, all destinations are located randomly in the network.

While creating these scenarios we keep in mind the real life cases where we might want to send a message to a single region or several different regions. We also add the random case to be able to see the general behavior of the algorithms.

In our simulations, we do not guarantee the connectivity of the generated network graph, but while calculating the success rate we omit the unreachable destinations in order to make the results more accurate.

4.1. Results

4.1.1. Success Rate. In this section we report simulation results which show the success rate of the algorithms for three different types of scenarios.

In Figure 6 success rates for randomly located destinations for all three algorithms are shown. As seen in Figure 6, both LBM-D and LBM-MST perform much better than PBM. Success rates for LBM-D and LBM-MST are close to each other, and while the network is getting denser, they reach success rates close to 1. An important point is that for sparse networks success rate of PBM is very low compared to LBM-D and LBM-MST. In addition, it appears LBM-D performs slightly better than LBM-MST for randomly deployed networks.

Figure 7 shows the success rates of the algorithms for randomly located destinations close to each other, in other words for destinations located in the same subregion. Our first observation is that, generally for destinations in the same region, results are better than the case where destinations are randomly distributed. Especially for sparse networks this observation holds. In this case, we see that PBM performs slightly better than LBM-MST. LBM-MST performs better than LBM-D for sparser networks. However, while the networks get denser, LBM-D starts to perform better than LBM-MST and reaches to the success rate of PBM. For

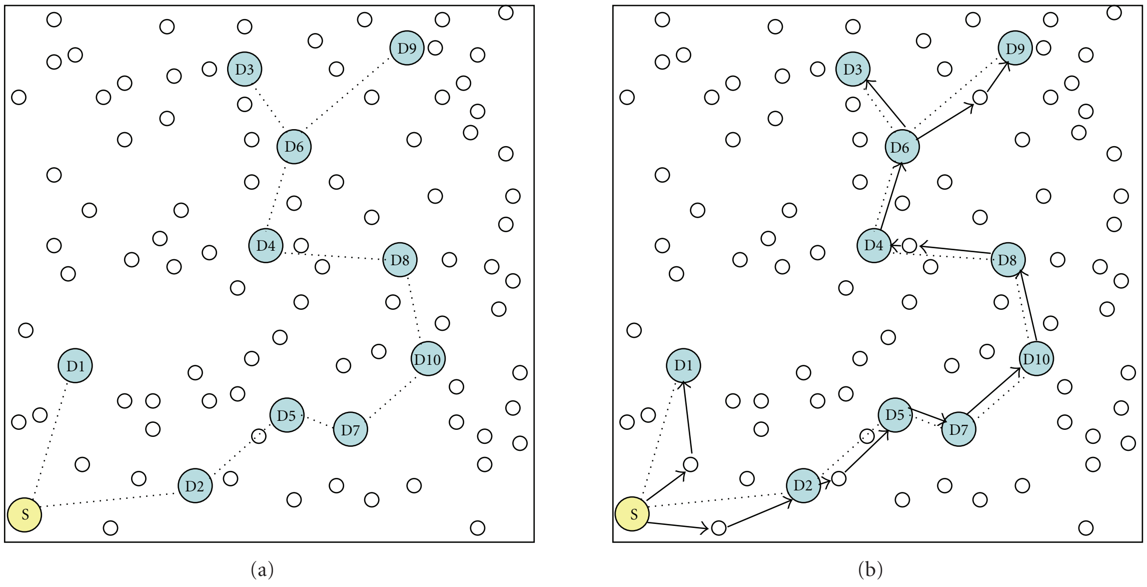


Figure 5: A sample run of LBM-MST with 10 destinations.

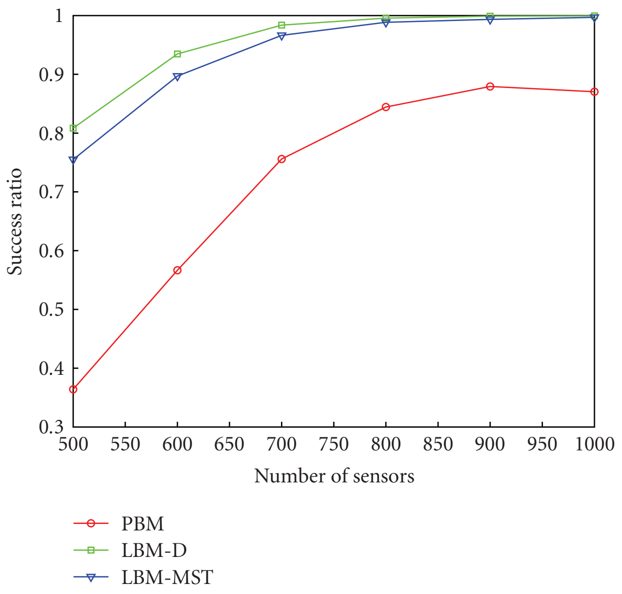


Figure 6: Success rates for randomly located destinations.

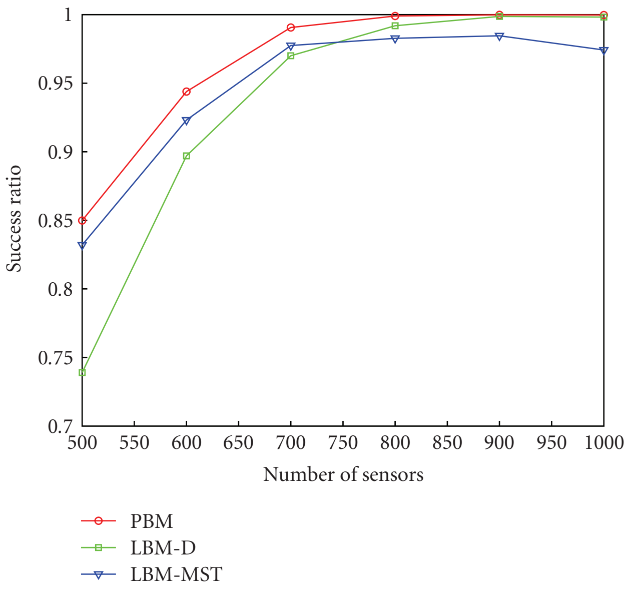


Figure 7: Success rates for destinations grouped together.

networks with number of sensors greater than 700, all three algorithms achieve high success rates which are close to 1.

Success rates of the algorithms for randomly deployed destinations to separated subregions are shown in Figure 8. PBM and LBM-D perform better than the LBM-MST, especially for sparse networks that have less than 800 nodes. The main reason of low success rate of LBM-MST for sparse networks is its inability of reaching child nodes of a node when unable to reach the node itself. However, in denser network environments, the probability of not reaching a node is low, so LBM-MST performs as well as PBM and LBM-D. We can observe that PBM and LBM-D achieve success rates very close to each other.

4.1.2. End-to-End Delay. In this section we report simulation results for average end-to-end delay, that is the time passes from the transmission of a message from the sink until the reception by a destination node. As mentioned earlier, we give the end-to-end delays in terms of number of hops. Therefore, we omit the processing time that is spent in sensor nodes during the routing. Indeed this approach favors the PBM, because it spends much more time for message processing than LBM-D or LBM-MST.

Results of experiments for average end-to-end delay for randomly located destinations are shown in Figure 9. We observe that PBM and LBM-MST yield high average end-to-end delays than the LBM-D. We can also say that average

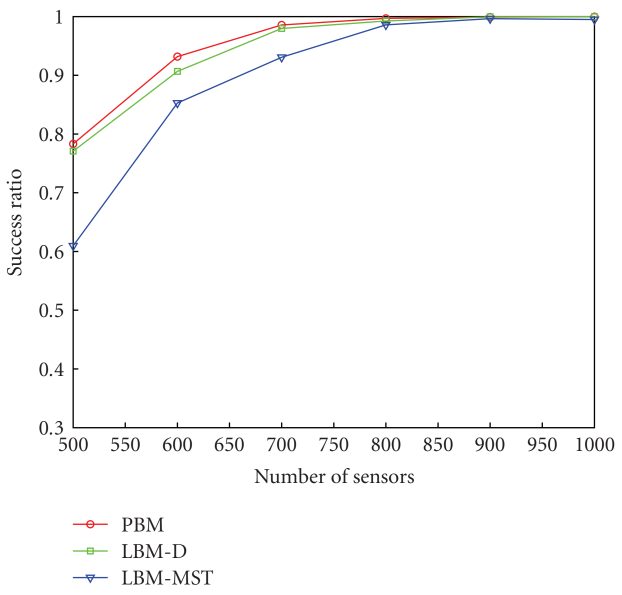


FIGURE 8: Success rates for separately distributed destinations.

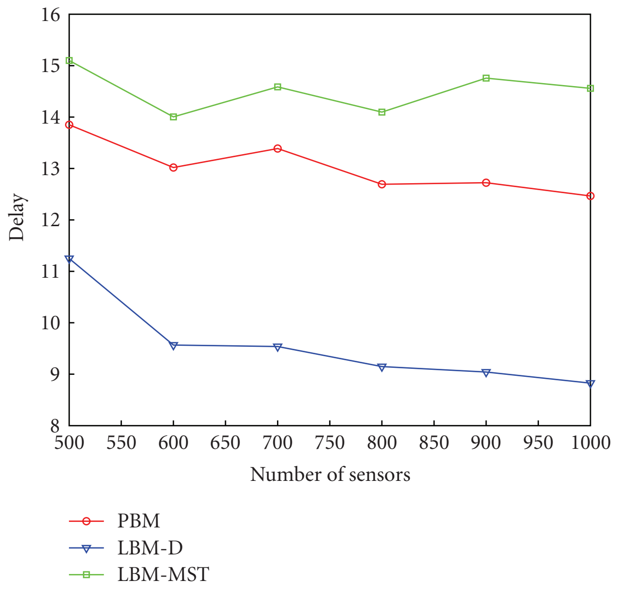


FIGURE 9: Average end-to-end delay for randomly located destinations.

end-to-end delays for PBM and LBM-MST are not directly affected with the increasing number of sensors deployed. They follow a path which is close to a straight line. High end-to-end delays occur in PBM and LBM-MST, because they tend to branch later than LBM-D. Especially for LBM-MST, messages have to traverse the minimum spanning tree to reach the destinations which increases the end-to-end delays. Figure 9 also shows that average end-to-end delay for LBM-D decreases while the number of sensors increases.

Figure 10 shows the average end-to-end delays of the algorithms for randomly located destinations that are close to each other. In this case, again PBM and LBM-D perform better than LBM-MST for the same reasons mentioned

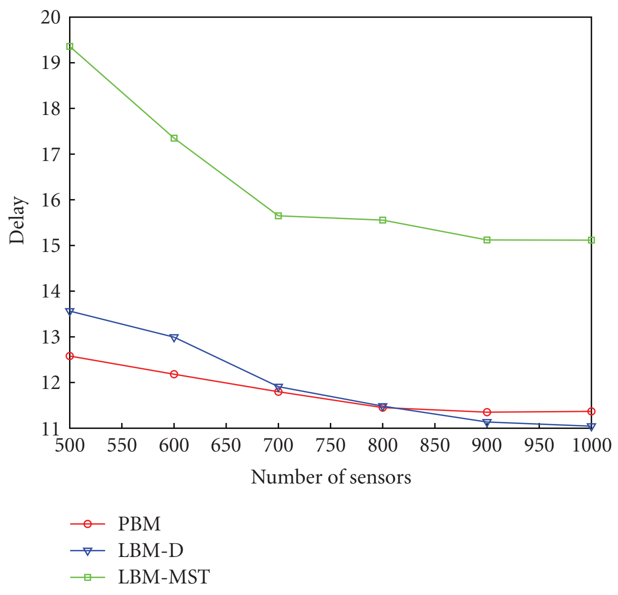


FIGURE 10: Average end-to-end delay for destinations grouped together.

before. For all three algorithms, average end-to-end delays decrease while the networks get denser.

In Figure 11, simulation results showing average end-to-end delay for randomly located destinations into separated regions are given. LBM-MST results with higher end-to-end delays than PBM and LBM-D as expected. Average end-to-end delay for three algorithms do not change much with the number of sensors deployed but a very slight decrease can be observed while the networks get denser.

4.1.3. Number of Total Transmissions. In this section we compare the power consumptions of the algorithms in terms of total number of transmissions when a multicast message is delivered to all destinations. Indeed total power consumption is the sum of energy spent during receptions and transmissions and total power used for processing in sensors. Ideally we should also consider the case when the sensor nodes are idle or in sleeping mode. However, these parameters are hardware specific and some of them can be neglected when compared to transmission power. Therefore, in order to have an idea of power consumption of the algorithms, we just compare the total number of transmissions needed to deliver a message to all destinations.

Figure 12 shows the comparison of power consumptions of the three algorithms in terms of number of total transmissions for randomly located destinations. We see that LBM-MST does less transmissions than both LBM-D and PBM, especially for denser networks. Number of transmissions for PBM is close to LBM-MST but it increases faster while the number of sensors in the network is increasing. In addition, if we consider the processing power required by PBM, we can easily say that LBM-MST performs much better than PBM in terms of power consumption because PBM does 2^n comparisons at a node while LBM-MST does n comparisons, where n is the number of neighbors, in order to decide

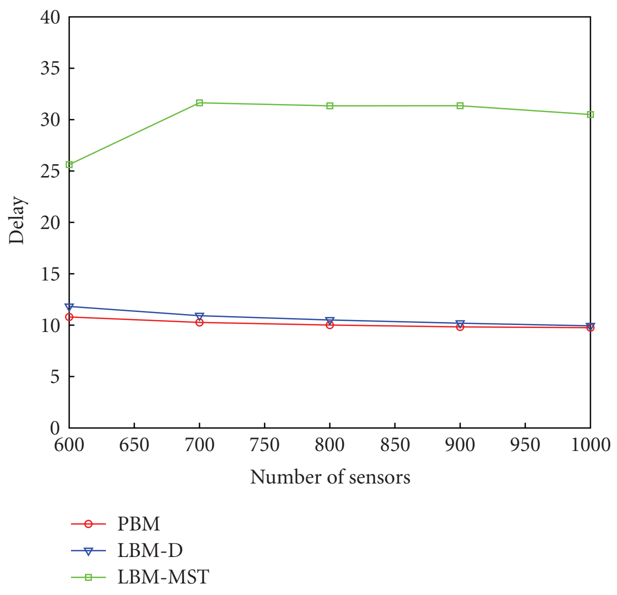


Figure 11: Average end-to-end delay for separately distributed destinations.

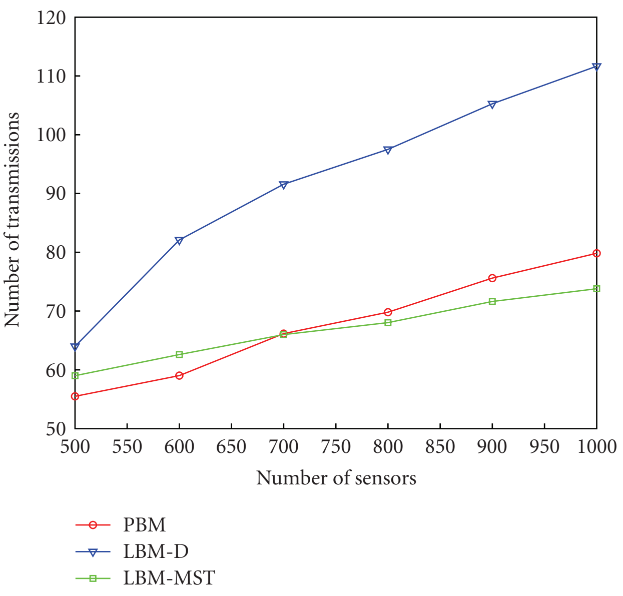


Figure 12: Number of transmissions for randomly located destinations.

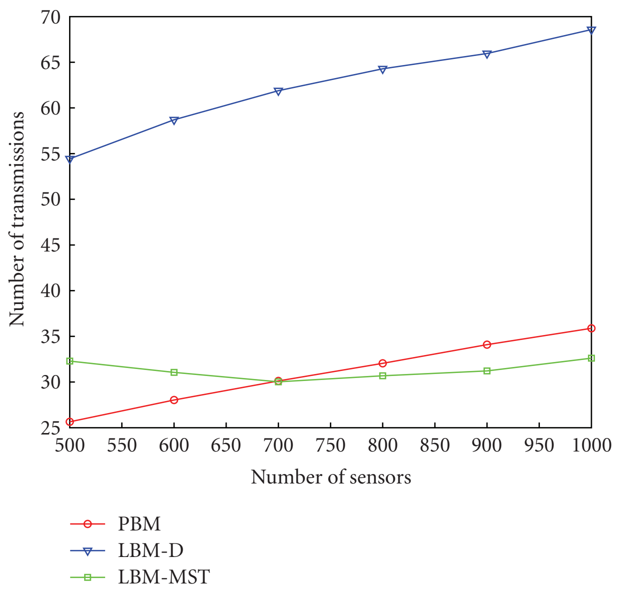


Figure 13: Number of transmissions for destinations grouped together.

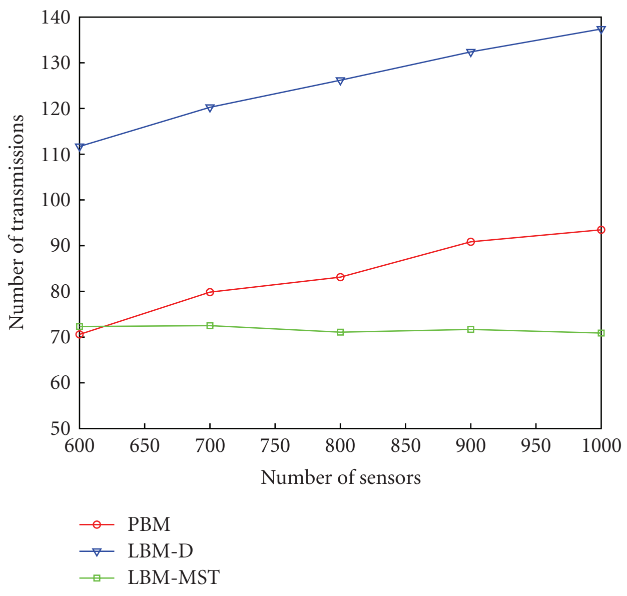


Figure 14: Number of transmissions for separately distributed destinations.

the neighbors which will get the message next. Additionally, LBM-D needs much more transmissions than LBM-MST and PBM in order to deliver the messages to all destinations, therefore it consumes much more power than LBM-MST and PBM.

Total number of transmissions for randomly located destinations in the same region are given in Figure 13. When all the destinations are located close to each other, total number of transmissions decreases, as expected. The difference between LBM-D and other algorithms becomes greater in this case. Number of transmissions made by PBM grows linearly while the network gets denser; on the other hand the increase in LBM-MST is slower, and total number of transmissions are smaller than PBM for denser networks.

The results of experiments performed in order to compare the total number of transmissions of each algorithm for randomly located destinations into separated regions are given in Figure 14. When the destinations are distributed to separated locations, LBM-MST uses less transmissions than PBM and LBM-D. An important difference arises between LBM-MST and PBM in this case. While the network is getting denser, number of transmissions for PBM grows rapidly and linearly, on the other hand number of transmissions for LBM-MST becomes almost constant.

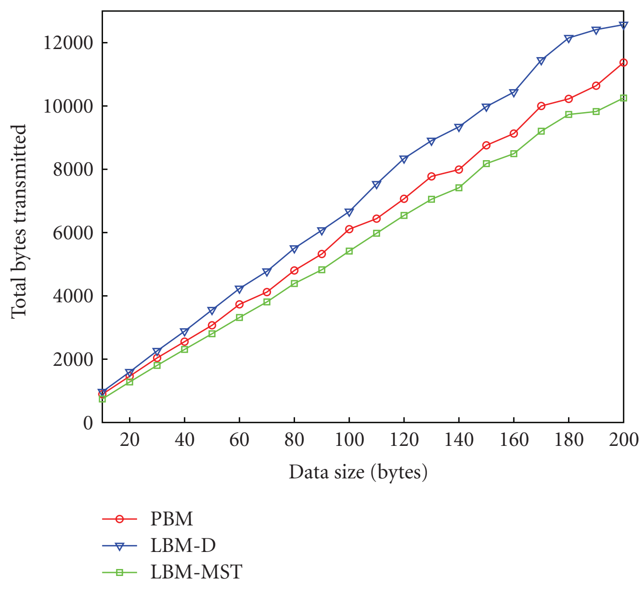


Figure 15: Traffic overhead versus data size.

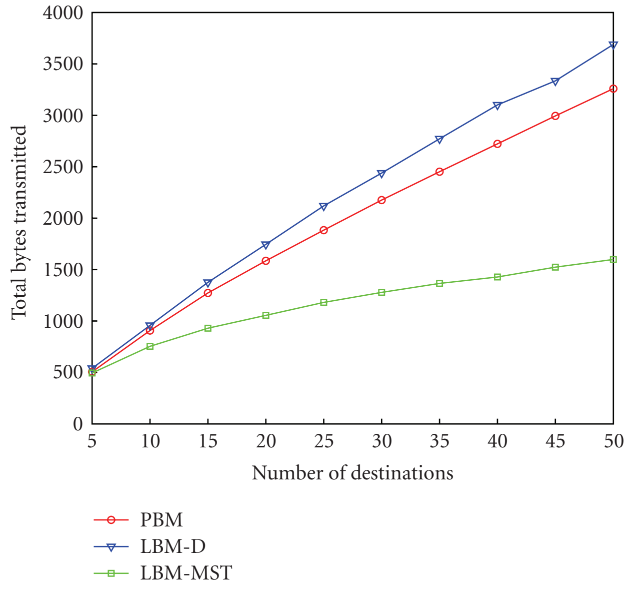


Figure 16: Traffic overhead versus number of destinations.

4.1.4. Total Traffic. In this section we evaluate the total traffic imposed by each algorithm in terms of total bytes transmitted in order to deliver all multicast messages to destinations. This analysis is important because size of multicast packets also affect the power consumption. For PBM and LBM-D, multicast messages have the same size when the number of destinations are the same. However, for LBM-MST we should also include the pointers that indicate the parent of a destination in the Euclidean minimum spanning tree. In order to see the effect of these pointers, we set up a scenario where 1000 sensor nodes are deployed randomly and we should deliver multicast messages to 10 destinations. We assume that a multicast message is composed of user data, destination node coordinates and pointers for parent destinations. For PBM and LBM-D, bytes allocated for pointers to parent destinations will be 0. We allocate 4 bytes for each destination node coordinate and 1 byte for each pointer. We also allocate 4 bytes for the next node coordinates. User data size changes from 10 bytes to 200 bytes and its affect on traffic overhead is analyzed. The result of this experiment is shown in Figure 15.

As seen in Figure 15, total traffic increases linearly while the actual data size grows for each algorithm. LBM-MST causes the minimum total traffic among the three algorithms although it uses larger multicast packets to hold extra pointers to parent destination nodes of Euclidean minimum spanning tree, because LBM-MST makes less transmissions than LBM-D and PBM to deliver the multicast messages to all destinations. We observe that results are parallel to the total number of transmissions, which means that extra pointers in LBM-MST packets do not significantly affect the power efficiency of LBM-MST.

We conduct another experiment to see the effect of increasing number of destinations to total traffic (in bytes) transmitted by each algorithm. In order to observe this effect, we keep the actual data size constant at 10 bytes. This is a small data size which is comparable with the size of destination information and the size of pointers. In this scenario, 1000 sensor nodes are randomly deployed and multicast messages should be delivered to destinations whose numbers vary from 5 to 50. Figure 16 depicts the result of this experiment.

Figure 16 shows that total traffic caused by LBM-D and PBM algorithms increase approximately with the same rate while the number of destinations increases. Despite the extra pointers hold in LBM-MST packets, total traffic caused by LBM-MST is less than the other algorithms and also it increases with a slower rate. For large number of destinations, efficiency of LBM-MST gets clearer as seen in Figure 16. For example, when number of destinations are greater than 40, total traffic caused by LBM-MST is almost half of the traffic caused by other algorithms.

5. Conclusion and Future Work

In this paper we propose two location-based multicast routing algorithms for wireless sensor networks, namely LBM-D and LBM-MST. Our algorithms aim to deliver the multicast messages to destinations in a distributed and scalable fashion. In addition, by reducing the number of transmissions, they conserve energy which is a scarce resource for sensor nodes.

The first algorithm we propose is LBM-D, which basically groups the destinations according to the angles they make with the sender node. For each group of destinations a message is forwarded via an appropriate neighbor which is selected with our neighbor selection scheme. This approach decreases the number of transmissions made, by following a common path for destinations that reside at same direction. LBM-MST is our second algorithm which first calculates an Euclidean Minimum Spanning Tree and forces LBM-D to follow the paths of this MST. This algorithm decreases the number of branching in multicast trees, so less transmissions are required to deliver the multicast messages to destinations.

After having designed our algorithms, we evaluated them with a custom simulator we developed. We implemented the PBM [9] algorithm and our algorithms in this simulation environment. The simulation results show that LBM-D and LBM-MST achieve success rates as good as PBM does, especially in dense networks. Since PBM is not a scalable algorithm, which needs to make 2^n comparisons for neighbor selection among n neighbors, LBM-D and LBM-MST can be used instead of it, because both of them use a scalable neighbor selection approach. In addition, especially LBM-MST requires less transmissions than PBM, so it stands as a better solution for multicasting in sensor networks where energy conservation is essential.

In conclusion, we can say that LBM-D and LBM-MST algorithms are good candidates to be used in location-based multicast routing protocols for wireless sensor networks.

Acknowledgments

The authors thank the European Commission for partially supporting this work through the FP7 framework project NEWCOM++, Network of Excellence in Wireless COMmunications++, Contract no. 216715. They also thank the IBM Corporation for partially supporting this work through IBM Faculty Award to the second author.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] C. S. R. Murthy and B. S. Manoj, *Ad Hoc Wireless Networks, Architectures and Protocols*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2004.

[3] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 22–31, Atlanta, Ga, USA, September 2002.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 56–67, Boston, Mass, USA, August 2000.

[5] W. Heinzelman, J. Kulik, and 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*, pp. 174–185, Seattle, Wash, USA, August 1999.

[6] S. Lindsey and C. Raghavendra, "PEGASIS: power-efficient gathering in sensor information systems," in *Proceedings of IEEE Aerospace Conference*, vol. 3, pp. 1125–1130, Big Sky, Mont, USA, March 2002.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS '00)*, p. 110, Maui, Hawaii, USA, January 2000.

[8] H. Ö. Tan and I. Körpeoğlu, "Power efficient data gathering and aggregation in wireless sensor networks," *SIGMOD Record*, vol. 32, no. 4, pp. 66–71, 2003.

[9] M. Mauve, H. Fußler, J. Widmer, and T. Lang, "Mobihoc poster: position-based multicast routing for mobile ad-hoc networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 53–55, 2003.

[10] W. Zhang, X. Jia, C. Huang, and Y. Yang, "Energy-aware location-aided multicast routing in sensor networks," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WCNM '05)*, vol. 2, pp. 901–904, Wuhan, China, September 2005.

[11] M. Transier, H. Fußler, J. Widmer, M. Mauve, and W. Effelsberg, "Scalable position-based multicast for mobile ad-hoc networks," in *Proceedings of the 1st International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWim '04)*, San Jose, Calif, USA, October 2004.

[12] S. Basagni, I. Chlamtac, and V. R. Syrotiuk, "Location aware, dependable multicast for mobile ad hoc networks," *Computer Networks*, vol. 36, no. 5-6, pp. 659–670, 2001.

[13] J. A. Sanchez, P. M. Ruiz, and I. Stojmenovic, "Energy-efficient geographic multicast routing for sensor and actuator networks," *Computer Communications*, vol. 30, no. 13, pp. 2519–2531, 2007.

[14] S. Wu and K. Candan, "GMP: distributed geographic multicast routing in wireless sensor networks," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, p. 49, Lisboa, Portugal, July 2006.

[15] R. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.