LOCATION BASED MULTICAST ROUTING ALGORITHMS FOR WIRELESS SENSOR NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Hakkı BAĞCI

August, 2007

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. İbrahim Körpeoğlu(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Dr. Cengiz Çelik

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Adnan Yazıcı

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

LOCATION BASED MULTICAST ROUTING ALGORITHMS FOR WIRELESS SENSOR NETWORKS

Hakkı BAĞCI
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. İbrahim Körpeoğlu
August, 2007

Multicast routing protocols in wireless sensor networks are required for sending the same message to multiple different destination nodes. Since most of the time it is not convenient to identify the sensors in a network by a unique id, using the location information to identify the nodes and sending messages to the target locations seems to be a better approach. In this thesis 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 sends a 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 target nodes. According to the calculated MST, multicast message is forwarded to destination nodes. This helps reducing the total energy consumed for delivering the message to all target nodes since it tries to minimize the number of transmissions. We compare these two algorithms with each other and also against another location based multicast routing protocol called PBM according to success ratio in delivery, number of total transmissions, traffic overhead and average end to end delay metrics. The results show that algorithms we propose are more scalable and energy efficient, so they are good candidates to be used for multicasting in wireless sensor networks.

Keywords: Wireless Sensor Networks, Location Based Multicasting, Geographic Routing.

ÖZET

KABLOSUZ ALGILAYICI AĞLAR İÇİN KONUM BAZLI ÇOKLU GÖNDERİM YOL BULMA ALGORİTMALARI

Hakkı BAĞCI
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. İbrahim Körpeoğlu
Ağustos, 2007

Kablosuz algılayıcı ağlarda bir mesajı birden fazla hedef algılayıcı düğüme göndermek için çoklu gönderim yol bulma protokollerine ihtiyaç vardır. Algılayıcı ağlarında çoğu zaman algılayıcı düğümlere birbirinden farklı belirleyiciler tahsis etmek etkili bir yol olmadığından, konum bilgisini ayırt edici özellik olarak kullanmak ve mesajları hedef konumlara göndermek daha iyi bir yaklaşım olarak görünüyor. Bu tezde kablosuz algılayıcı ağlar için algılayıcı düğümlerin konum bilgisini kullanarak çalışan, iki yeni dağıtık çoklu gönderim algoritması öneriyoruz. İlk algoritma çoklu gönderim ağacındaki toplam dal sayısını ve buna bağlı olarak toplam gönderme sayısını azaltmak için hedef düğümleri açısal konumlarına göre gruplayarak, her gruba bir mesaj gönderir. İkinci algoritma ise kaynak düğümde hedef düğümlerin konum bilgisini kullanarak Öklit minimum kaplama ağacı hesaplar. Çoklu gönderim mesajları, oluşturulan bu ağaca göre hedef düğümlere gönderilir. Bu yaklaşım toplam gönderim sayısını azaltmayı amaçladığından mesajları hedef düğümlere göndermek için kullanılan toplam enerji miktarının düşürülmesini sağlar. Bu iki algoritmayı birbiriyle ve başka bir konum bazlı çoklu gönderim protokolü olan PBM ile iletim başarısı, toplam gönderim sayısı, uçtan uca gecikme süresi ve gönderilen toplam data miktarı açısından karşılaştırdık. Sonuçlar gösterdi ki, önerdiğimiz algoritmalar daha ölçeklenebilir ve enerji kullanımı bakımından daha etkindir. Bu sebeple kablosuz algılayıcı ağlarda çoklu gönderimde kullanılmak için iyi birer adaydırlar.

Anahtar sözcükler: Kablosuz Algılayıcı Ağlar, Konum Bazlı Çoklu Gönderim, Coğrafi Yol Bulma.

To my wife, Figen...

Acknowledgement

I would like to express my deepest gratitude and profound respect to my supervisor Asst. Prof. Dr. İbrahim Körpeoğlu for his expert guidance and suggestions, positive approach throughout my masters study and his efforts and patience during supervision of this thesis.

I acknowledge with thanks and appreciation the jury members, Dr. Cengiz Çelik and Prof. Dr. Adnan Yazıcı for reviewing and evaluating my thesis.

I thank to the members of Networking and Systems Group at Bilkent University for their invaluable contributions to my studies during the years 2004 to 2007.

I also thank to TÜBİTAK UEKAE / ILTAREN for supporting my academic studies.

I thank to my wife, for her understanding and support during my thesis study.

Finally, I would like to express my thanks to my parents and brother for their love, trust and every kind of supports.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

By the last decade, low-cost tiny sensors 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 (WSN). 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 [13]. Intrusion detection of enemies, battlefield monitoring and early warning systems for detecting chemical attacks can be listed in military applications category. Environment monitoring includes pollution detection, habitat monitoring, forest fire detection applications. Health monitoring systems can be used for patients who need constant monitoring of their body temperature, heart rate and other data that can be measured by sensors. Sensor

networks enable building of smart homes which autonomously control the air temperature, lightning, etc. and remotely communicate with electronic equipments. In industry, sensor networks are used for monitoring warehouses, production lines and other many applications that we cannot account for all here. It is not hard to see that in near future application areas of sensor networks will broaden with the further development of micro-electro mechanical systems technology.

Wireless sensor networks have similarities to wireless ad hoc networks but there are essential differences between sensor networks and conventional ad hoc networks. Primary differences can be listed as follows [14]:

- Number of nodes in sensor networks are much more than that of ad hoc networks. Thousands of nodes can be deployed in a certain area to form a sensor network. Usually number of nodes in ad hoc networks is in the order of tens or hundreds.
- Sensor nodes are strictly energy constrained. They have batteries with small capacities which are usually not replaceable. On the other hand, in conventional ad hoc networks, nodes have large capacity batteries or connected to a power source.
- Usually assigning unique global identifiers to sensor nodes is not feasible nor required, because routing in sensor networks is data-centric, whereas in ad hoc networks, routing is address-centric which requires unique addressing.
- Sensor nodes have limited processing power and memory, so most of the algorithms designed for ad hoc networks cannot be used directly in sensor networks.

Due to their special requirements, wireless sensor networks need different mechanisms for routing [13]. Routing algorithms for wireless sensor networks should share the following properties:

- Usually it is not feasible to make intervention after deployment, so algorithms for setup should be autonomous.

- Nodes in sensor networks have only local knowledge due to lack of an infrastructure, so routing algorithms should be distributed.
- Since the number of nodes can be very high and networks can be dense, routing algorithms should be scalable.
- Sensor nodes have limited power, so routing algorithms should be energy aware.

Routing protocols for wireless sensor networks aims to solve 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 in wireless sensor networks. Flooding and gossiping are two primitive approaches which are not suitable for most of the sensor network applications because they do not have energy conservation mechanisms. Other well-known data dissemination protocols are rumor routing [4], directed diffusion [7], and SPIN [6]. In rumor routing, path creation and maintenance are done by agents which are packets generated by nodes. These packets travel in the network and update nodes' routing tables. Directed diffusion makes use of interest gradients where interests are defined by one or more attributes. Basically, a sink node publishes its interest and data related to the interest flows along the reverse path to the sink. SPIN consists of a set of protocols named sensor protocols for information via negotiation. Instead of sending raw data, SPIN sends meta data to reduce power consumption. SPIN uses three types of messages; namely ADV, REQ and DATA. A sensor node publishes its data by sending an ADV message that contains the meta data of the real data, which is shorter in size. Any sensor node who requires the advertised data sends a REQ message to the advertising node. Then, actual data is sent with DATA messages.

Data gathering protocols aim to collect data from the network in efficient ways. PEGASIS [10], power-efficient gathering for sensor information systems, is

one of the well-known data gathering protocols. It assumes overall topology of the network is known to all sensor nodes. A chain of sensor nodes is constructed by starting from the furthest node to base station. The chain grows by adding the nearest neighbor until reaching a leader node which transmits the collected data to the base station. PEGASIS makes use of data aggregation, which prevents increasing of data size while proceeding on the chain. Binary scheme and Chain-Based Three-Level Scheme described in [11] are also data gathering protocols which are based on mainly a chain construction approach like PEGASIS. Binary scheme is used if nodes communicate by CDMA, and it is possible to use Chain-Based Three-Level Scheme when CDMA communication is not possible.

In our study, we focus mainly on a specific type of data dissemination problem. Given a set of destination nodes, we try to deliver a message to all these destinations. This is called multicasting. Most of the data dissemination protocols mentioned above can be classified as multicasting protocols indeed. However, they do not solve the problem we attack in this thesis. Some multicasting protocols related to our work are described briefly in Section 2.2.

In this thesis, 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 are Location Based Multicasting with Direction (LBM-D) and Location Based Multicasting according to Minimum Spanning Tree (LBM-MST). As usual for most location based routing algorithms, we assume that all nodes in the network know their own location 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 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) [12]. 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 contributions of this thesis can be summarized as follows:

- We propose two distributed location based multicasting algorithms (LBM-D and LBM-MST) for wireless sensor networks, which are power efficient and scalable.
- We develop a destination assignment approach for PBM [12] algorithm.
- We implemented a simple wireless sensor network simulator which makes the design and evaluation of the algorithms easier via its visual support.
- We evaluate LBM-D and LBM-MST by comparing them with each other and an existing location based multicasting algorithm called PBM.

The remainder of this thesis is organized as follows. In the next chapter, we describe some of the work related to location based routing and multicasting. In Chapter 3, we first give a detailed description of PBM [12] and how we have implemented it. Then in the same chapter, we describe our proposed algorithms in detail. In Chapter 4, we report the evaluation results of the algorithms, and we conclude the thesis in Chapter 5.

Chapter 2

Related Work

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

2.1 Location Based Routing for Ad Hoc and Sensor Networks

Location based routing algorithms mostly depend on the assumption that all nodes in the network know their own positions. Generally, each node learns its location information from GPS or any other location information service. The inexpensiveness of location information services and rapidly developing location information technology has increased the value of location-based routing [16]. Especially for mobile ad hoc networks where a static topology is not available to route the information through nodes, location based routing constitutes a good alternative.

Basically, in location based routing algorithms, the sender node knows the location information of the destination node(s) and sends this information with the actual data to the appropriate neighbor(s). Usually neighbor selection approaches are the parts of the algorithms which create the difference with other

location based routing algorithms. Mostly, location based routing algorithms adopt a greedy approach to find the neighbor(s) to forward the messages next.

Various location based routing algorithms have been developed with different neighbor selection approaches.

The algorithm proposed by Takagi and Kleinrock draws a straight line between the sender node and the destination node, and then it chooses the node among the neighbors of the sender which makes the maximum progress when projected onto this line [19]. Another approach called The Most Forward within Radius (MFR) in [19] uses a static transmission radius. The approach works by finding the node within the transmission radius which gives the maximum progress.

Finn [5] has developed a greedy routing approach which is based on the geographic distance between nodes. The current node in the network chooses the closest neighbor to the destination and transmits the message to it. A further example of this routing approach is Geographic Distance Routing (GEDIR) [17]. In this algorithm, among all the neighbors, if the current node decides that the best solution is to turn back to the previous node, then the message is dropped.

Kranakis, Singh and Urritia proposed the compass routing algorithm [9] (also known as DIR approach). This algorithm draws a straight line between the current node and the destination node. It works by finding the node which makes the smallest angle with this line. The node having the smallest angle is selected to forward the message next.

Bose, Marin, Stojmenovic and Urrutia described the *face* and Greedy-FACE-Greedy routing schemes which are stateless, and guarantee the delivery [3]. These schemes are improved by using two-hop neighborhood information and dominating set concept [18].

Geographic routing algorithm (GRA) proposed by Jain, Puri, and Sengupta holds routes to certain destinations [8]. In order to forward message GRA uses a greedy approach. However, routing tables need to be updated when the information in routing tables become invalid. Route discovery protocol used by [8]

finds a path from sink to destination and updates the routing tables of the nodes on this path. After route discovery, the message is sent from sink to destination over this path.

2.2 Location Based Multicasting for Ad Hoc and Sensor Networks

Multicasting 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.

In this section we briefly mention the previous studies about location based multicasting that are related to our study. Before describing the related work, we want to make emphasis on the distinction between geocasting and multicasting. Most of the work we found in the literature is about geocasting, which is a special type of multicasting in which destinations are located within the same region of the network.

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 [21]. 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 [21] 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-base forwarding area multicast tree (CoFAM) method, a cone-based forwarding area is defined from the sing to the given region. Only nodes in the forwarding area forwards 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 [12]. 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 [20], 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 [2] 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 [20], 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 [22]. 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.

Chapter 3

Location Based Multicasting Algorithms

In this chapter 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. [12] 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 hand-shake 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 :
$MessageID + RepLoc_{destination_1} + RepLoc_{destination_2} + ... + RepLoc_{destination_n}$.

In each node we hold a list of sent $MessageSignature$s abbreviated as SMS. 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* structure is identical to the hashtable structure whose asymptotic access time is $O(1)$.

3.2 Position Based Multicasting (PBM)

In this section, we describe Position Based Multicasting(PBM) algorithm proposed by Mauve et al. [12] 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:

$$f(w) = \lambda \frac{|w|}{|N|} + (1 - \lambda) \frac{\sum_{z \in Z} min_{m \in w}(d(m, z))}{\sum_{z \in Z}(d(k, z))} \tag{3.1}$$

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 the Expression 3.1 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 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 the Expression 3.1 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 3.1. 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 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 is 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.

```
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_destinations contains this then
7:     Remove this from M_destinations
8:     if M_destinations is empty then
9:       Continue for the next message
10:    end if
11:  end if
12:  Next_Nodes ← FindNextNodes(M_destinations)
13:  Assignments ← AssignDestNodes(M_destinations,Next_Nodes)
14:  for all assignment ASGN in Assignments do
15:    newMessage ← 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 3.1: Greedy Multicast Forwarding in PBM.

Pseudo-code for the procedure that finds the nodes to forward the message next is given in Algorithm 3.2. It takes the list of destinations as input to be used in cost calculations. First, all subsets of the neighbors of the current node is calculated. 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 the Expression 3.1.

Having found the subset with minimum cost, destinations are assigned to the nodes in the subset according to the procedure *AssignDestNodes* in Algorithm 3.3. The method used by PBM is not mentioned in [12], so we designed

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

Algorithm 3.2: Finding Next Nodes for Destinations in PBM.

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: Create an empty dictionary Assignments
2: Create a list uncoveredDestinations
3: Copy destinations to uncoveredDestinations
4: for all node n in nextNodes do
5:    nearestDestNode ← Find nearest destination node to n
6:    Add a new entry e to Assignments such that e_key is n and e_value is
      {nearestDestNode}
7:    Remove nearestDestNode from uncoveredDestinations
8: end for
9: for all node d in uncoveredDestinations do
10:   nearestNextNode ← Find nearest next node to d
11:   Add d into the list Assignments[nearestNextNode]
12:   Remove d from uncoveredDestinations
13: end for
14: return Assignments
```

Algorithm 3.3: Assignment of Destination Nodes to Next Nodes in PBM.

3.3 Our Proposed Algorithm I: 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 3.4 and sub-procedures used by LBM-D is given in Algorithms 3.5 and 3.6. We first explain the main flow of the algorithm shown in Algorithm 3.4 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, multicast messages are created for each group and sent to the corresponding next nodes. In addition, $MessageSignature$s of the sent messages are written into SMS cache of the current node.

```
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_destinations contains this then
7:     Remove this from M_destinations
8:   end if
9:   if M_destinations is not empty then
10:    node_group_list ← GroupDestinations(M_destinations)
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 Sent Message Signatures cache
15:      Forward newMessage to nextNode
16:    end for
17:  end if
18: end for
```

Algorithm 3.4: Location Based Multicasting with Direction.

Our approach for grouping the destinations according to the angles they make with the current node is implemented as seen in Algorithm 3.5. 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.

Overview of procedure *GroupDestinations* is given in Algorithm 3.5. It 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 3.5 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 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 3.6. The procedure *SelectNextNeighbor*

```
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 + α 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 3.5: Grouping Destinations in LBM-D.

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 3.6. 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.

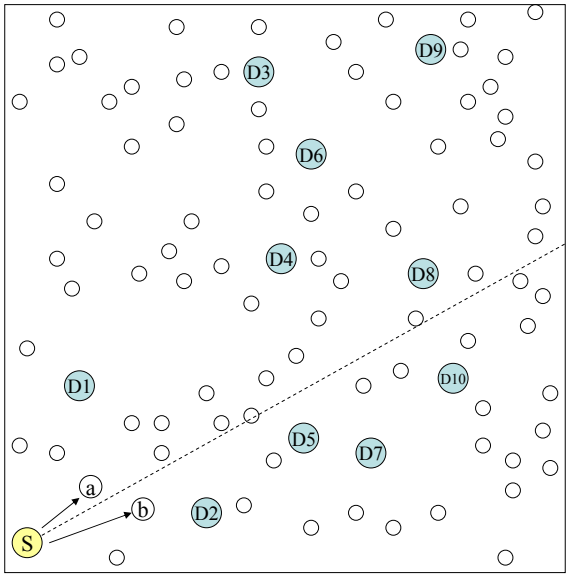
```
1: minNode ← null
2: minDiff ← infinity
3: currDiff ← infinity
4: for all neighbor n in this_neighbors do
5:    if destinations contains n then
6:       return n
7:    end if
8:    currAngle ← Calculate angle between this and n
9:    currDiff ← Calculate absolute value of (angle-currAngle)
10:   if minDiff is greater than currDiff then
11:      if n has more than 1 neighbor then
12:         minDiff ← currDiff
13:         minNode ← n
14:      end if
15:   end if
16: end for
17: return minNode
```

Algorithm 3.6: Selecting Next Neighbor in LBM-D.

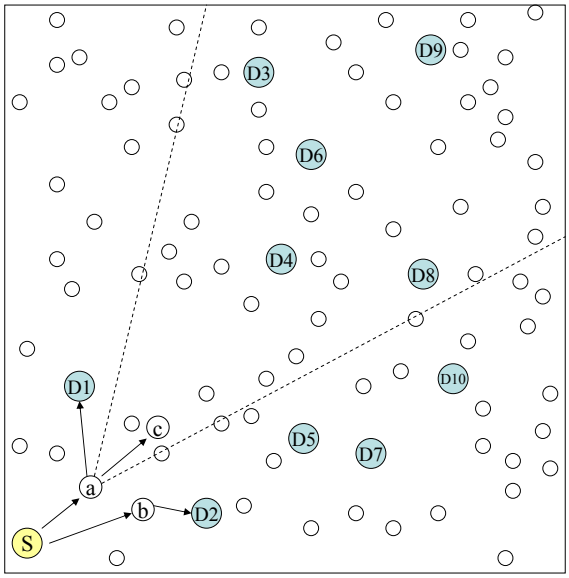
A sample run of LBM-D algorithm is given in Figures 3.1 and 3.2. A sample topology with a source node and 10 destinations (D1,...,D10) is shown in Figure 3.1(i). The dashed lines originating from the source node (depicted as S) divide the region into subregions according to the *alpha* which is used to group the

destinations. Solid arrows depict the transmission of multicast messages from one node to another. According to the case in Figure 3.1(i), 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 3.1(i) 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 b with target destinations (D2, D5, D7, D10). In Figure 3.1(ii), 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 3.6. On the other branch, node b forwards the message to D2 and destination D2 is reached. In Figure 3.2(iii), 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 3.2(iv) shows the final routes taken by the LBM-D algorithm in order to deliver the multicast messages to all destinations in the same manner explained above.

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.

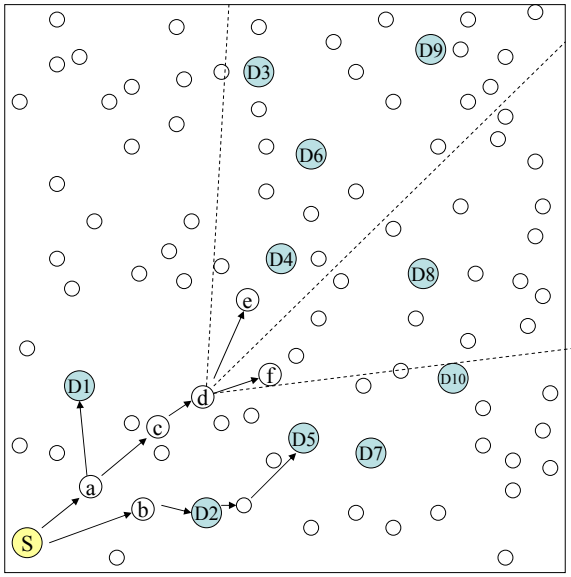


(i)

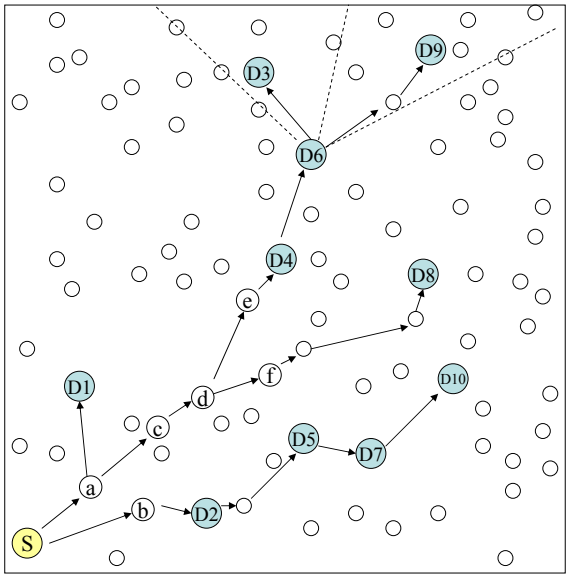


(ii)

Figure 3.1: LBM-D sample run.



(iii)



(iv)

Figure 3.2: LBM-D sample run cont'd.

3.4 Our Proposed Algorithm II: 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 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 carry is 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 4.3.4.

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 3.7.

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 3.3, for a multicast message that is sent from node S to D1, destination D1 is the next destination

```
1: for all received message M do
2:   if M_signature exists in Sent Message Signatures 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_nextDestinations contains this then
11:    Remove this from M_nextDestinations
12:  end if
13:  if M_allDestinations contains this then
14:    Remove this from M_allDestinations
15:    M_nextDestinations ← GetImmediateChildren(M_allDestinations)
16:  end if
17:  node_group_list ← GroupDestinations(M_nextDestinations)
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 Sent Message Signatures cache
22:    Forward newMessage to nextNode
23:  end for
24: end for
```

Algorithm 3.7: Location Based Multicasting with Direction According to MST.

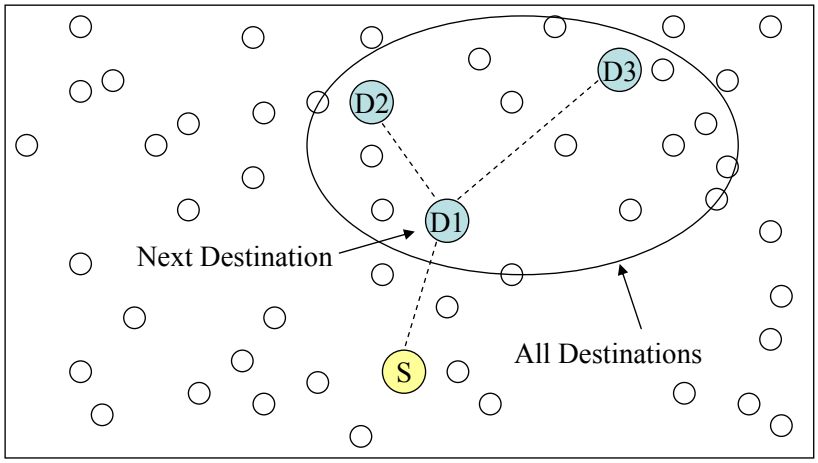


Figure 3.3: Next Destination and All Destinations.

where all destinations are D1, D2 and D3.

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 $M_{allDestinations}$. Then next destinations $M_{nextDestinations}$ are calculated by the procedure $GetImmediateChildren$. The first message is forwarded to next destinations with LBM-D. $M_{nextDestinations}$ 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.

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 $M_{allDestinations}$. This is achieved by holding a reference to the parent node of each destination in $M_{allDestinations}$ as shown in Figure 3.4. 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.

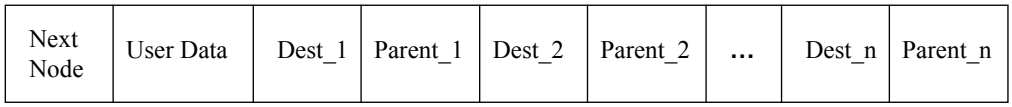


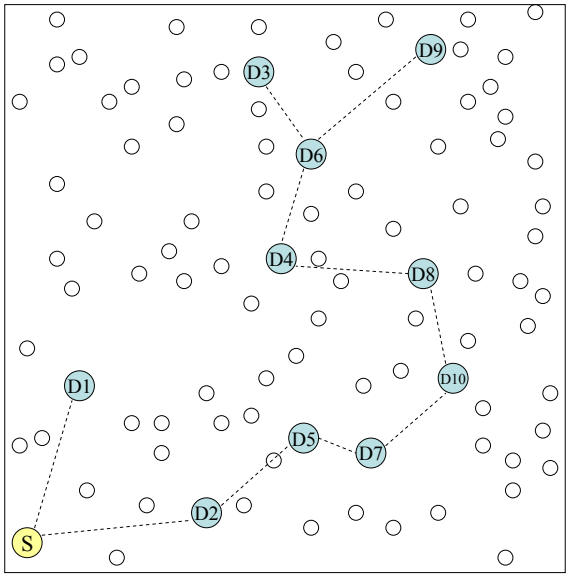
Figure 3.4: Multicast message used by LBM-MST.

```
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
```

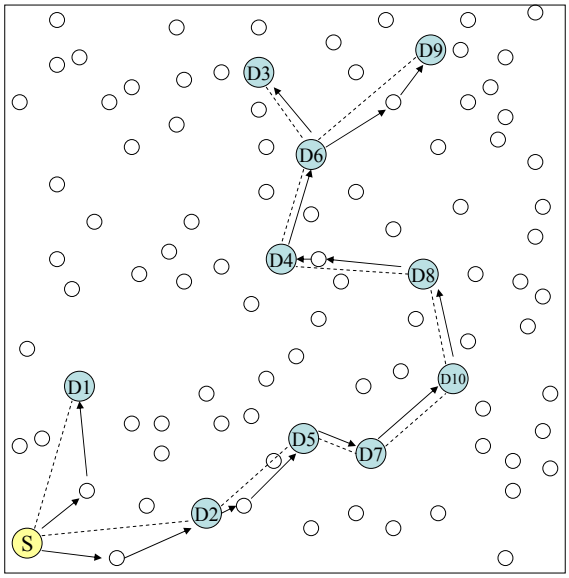
Algorithm 3.8: Finding Immediate Child Nodes in LBM-MST.

A sample run of LBM-MST algorithm is given in Figure 3.5. The topology of the network, which is the same with in Figure 3.1, is shown in Figure 3.5(i). The dashed lines in Figure 3.5(i) shows the Euclidean minimum spanning tree of 10 destinations. The solid arrows in Figure 3.5(ii) depict the actual paths followed by the LBM-MST algorithm. As seen from the Figure 3.5(ii), LBM-MST tries to follow the Euclidean MST to minimize the number of total transmissions. In this sample case, LBM-D makes 15 transmissions to deliver the multicast message to all destinations, whereas LBM-MST requires only 20.

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



(i)



(ii)

Figure 3.5: LBM-MST sample run.

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.

Chapter 4

Evaluation

In this chapter we report the results of the experiments done to evaluate our algorithms. We evaluate our algorithms by comparing them with PBM [12] algorithm and also with each other. In order to achieve this, we implemented a basic wireless sensor network simulator. On this simulator we implemented the PBM routing algorithm and our routing algorithms. Hundreds of sample runs show that our algorithms work as expected and they are feasible to be used as location based routing protocols in wireless sensor networks. Before presenting the experiment results, we first describe the simulator that we have developed.

4.1 Simulator

Simulation is a very powerful method to evaluate a new routing algorithm which is especially designed for wireless sensor networks. Preparing a real testbed would be very costly and time consuming since the number of sensor nodes are too large and repeating the experiments for many times is not feasible. Additionally, we do not define a complete protocol suite which can be deployed on real sensor nodes, but we offer algorithms that can be used in routing protocols. Therefore we preferred simulation rather than using a real testbed.

After having decided using simulation to evaluate our multicast routing algorithms, we had to make another decision on whether we should use an existing simulator or implement our own simulator. Implementing a new simulator is not an easy task; however, if the complexity of the simulator is well adjusted, it can be more useful than the existing ones. In addition, most of the existing network simulators do not support wireless sensor networks. Another good reason to write our own simulator is to avoid unnecessary details which are nothing to do with our work. We think that it is better to construct a strong theoretical base before going into the protocol details which are related to the deployment issues. Our aim was to see if the algorithms we offer are good enough to be used as the core part of a location based routing protocol for wireless sensor networks.

Another important issue is the visualization of the process that is going on during the routing in the network. It makes easier to analyze the behavior of the algorithms and to see the weaknesses. So we wanted to see all the paths that are constructed by the routing algorithms on the graph of sensor nodes.

Based on the reasons mentioned above we decided to implement a basic wireless sensor network simulator which is capable of performing the following tasks:

- Generating randomly distributed sensor network topologies,
- Running algorithms on densely deployed networks with high number of sensors,
- Repeating the experiments many times on different randomly generated topologies,
- Visually showing the constructed paths on the network,
- Generating customizable results beside the success ratio, number of total transmissions and end-to-end average delay.

We implemented our 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 can immediately see the output of the algorithm on the sensor network graph. For long runs we prefer writing code blocks in which the input parameters can be easily substituted. Numerical results are written on the formatted files which are used for generating graphical charts.

The general approach that we follow for evaluating our algorithms is summarized in Figure 4.1.

```
1: for all Simulation Scenarios S do
2:   for all Routing algorithms A (PBM, LBM-D, LBM-MST) do
3:     for seed = 1 to n do
4:       Generate random network topology T using seed
5:       Run experiment with A for S on T
6:       Store the output in a table
7:     end for
8:     Calculate the average values of the stored result
9:     Write the average values into the output file
10:  end for
11: end for
```

Algorithm 4.1: Simulation Approach.

This approach ensures that each algorithm is run on the same random topologies, so it prevents any biased results that may occur due to the specific topology of the network generated.

As summarized in Figure 4.1, for each scenario we run PBM, LBM-D and LBM-MST algorithms with different seeds, and for each experiment we output the results, namely success rate, number of total transmissions, end-to-end delay and traffic overhead, and for each algorithm we calculate the average values for these metrics.

4.2 Scenarios

We evaluated our algorithms by comparing with a similar location based multicast routing algorithm PBM [12]. 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.

In order to generate the same network topologies for all algorithms, we used a fixed random seed for all algorithms, but at each iteration this random seed is changed to obtain unbiased results.

We focus mainly on the following metrics to evaluate our algorithms:

- 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.
- 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 on the sensors. We also neglect the time spent for processing in the sensors.
- 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.
- 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 300x300 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 experiment 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. Indeed, in real applications we usually do not know the exact locations of any sensor in the region. Instead of sending messages to sensors, we send the messages to the regions or locations where we hope there exists some sensor nodes in those locations. This difference with real life and our simulation setup does not affect the general behavior of our algorithms. The only difference is we are sure that there exist a sensor node where we send the message in our simulations. To keep things simple, we just omit the regions and we directly send the messages to sensors whose locations are known. This approach can easily be applied to real life case by using a simple function that decides whether a sensor is inside the region that we want to send our message.

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 created three kind of simulation scenarios:

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. A sample deployment for this scenario is shown in Figure 4.1.

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

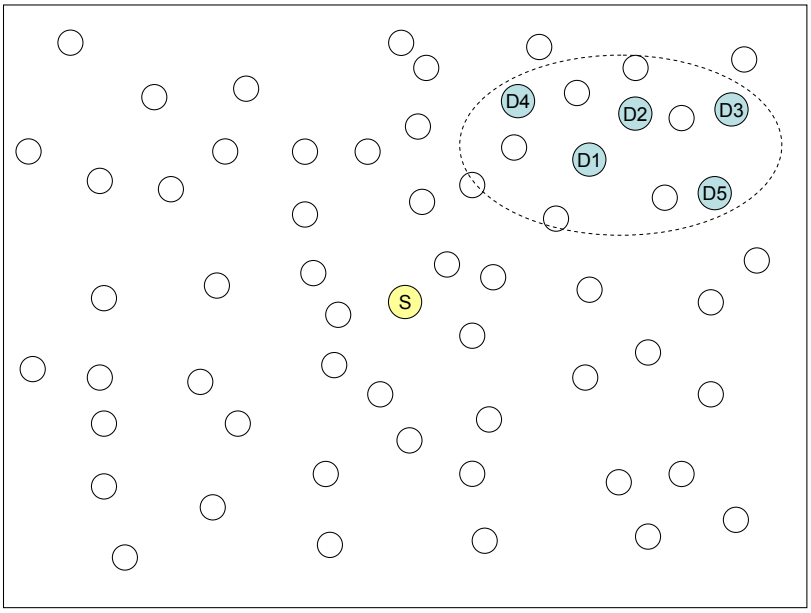


Figure 4.1: Destinations grouped together.

run, while changing the topology each time. A sample deployment for this scenario is shown in Figure 4.2.

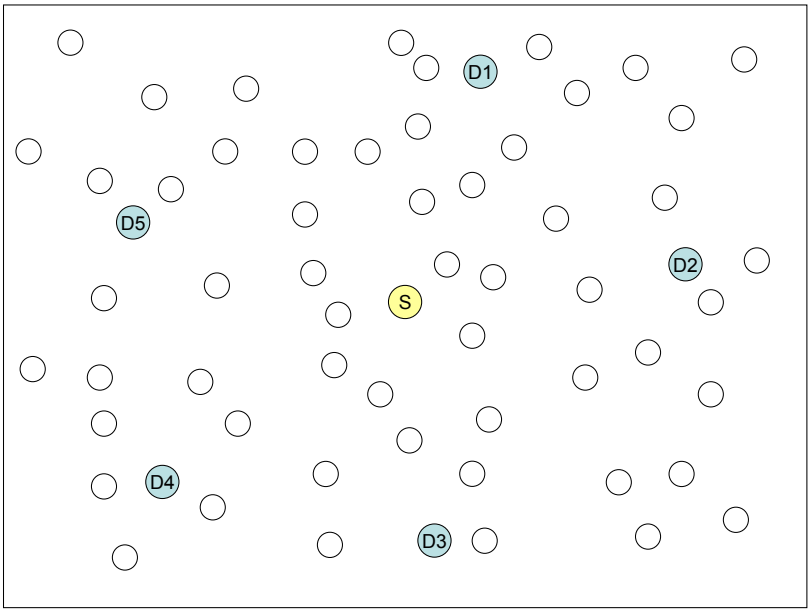


Figure 4.2: Destinations distributed separately.

3. Destinations are located randomly: In this scenario, all destinations are located randomly in the network. A sample deployment for this scenario is shown in Figure 4.3.

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.

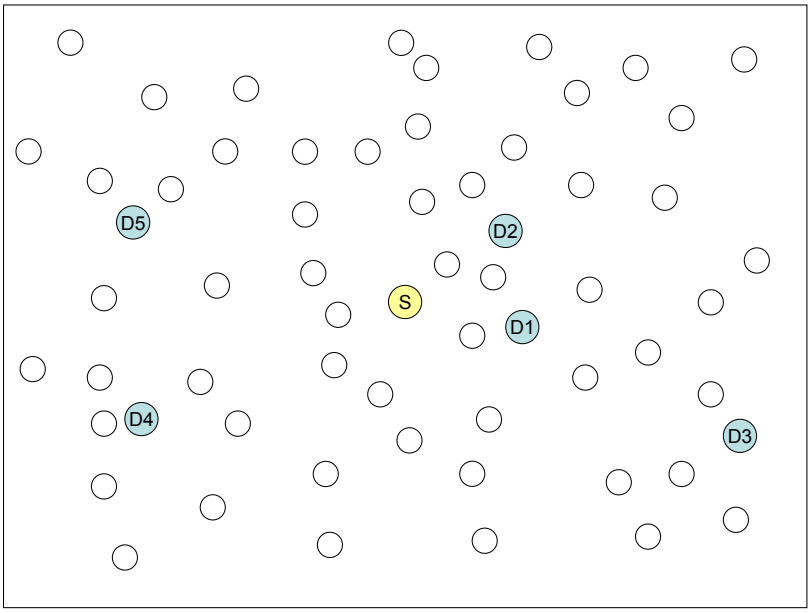


Figure 4.3: Destinations distributed randomly.

4.3 Results

4.3.1 Success Rate

In this section we report simulation results which show the success rate of the algorithms for three different type of scenarios.

In Figure 4.4 success rates for randomly located destinations for all three algorithms are shown. As seen in Figure 4.4, 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.

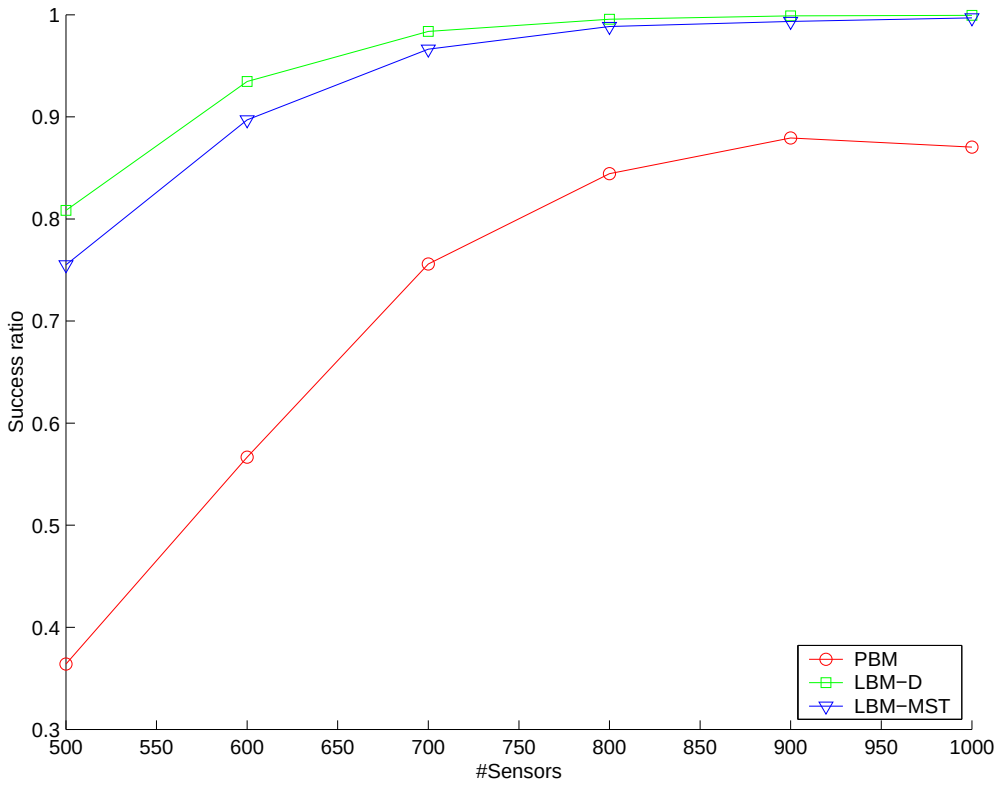


Figure 4.4: Success rates for randomly located destinations.

Figure 4.5 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 networks with number of sensors is greater than 700, all three algorithms achieve high success rates which are very close to 1. A slight decrease can be observed for LBM-MST after 900 sensors.

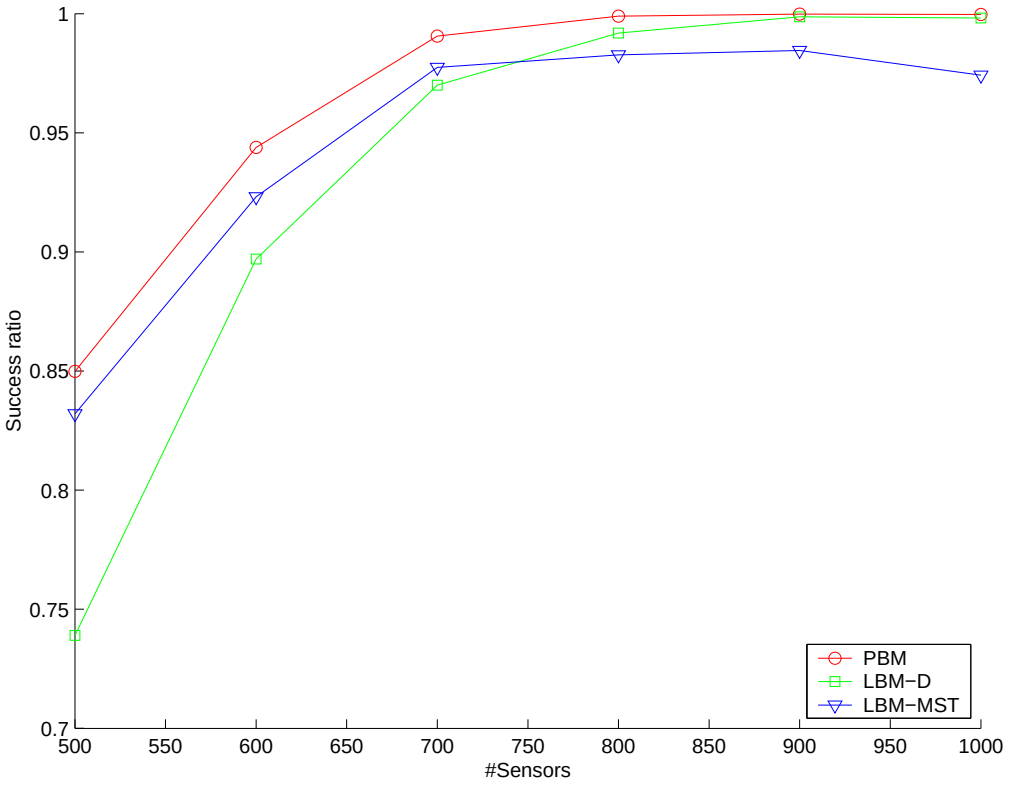


Figure 4.5: Success rates for destinations grouped together.

Success rates of the algorithms for randomly deployed destinations to separated subregions are shown in Figure 4.9. 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.

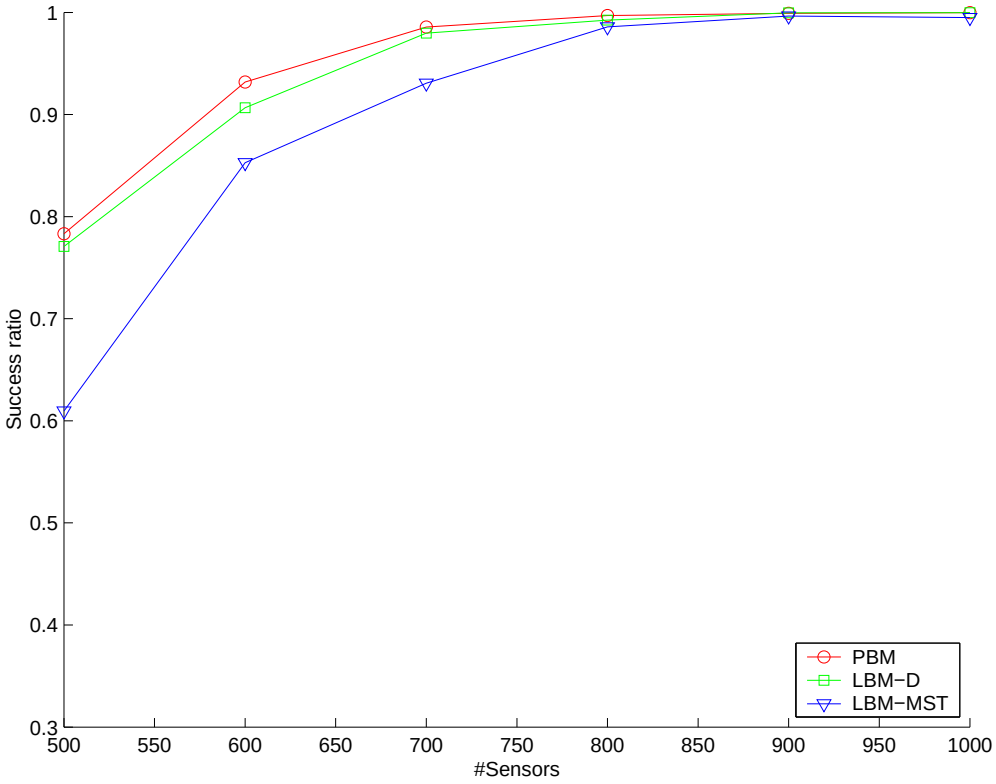


Figure 4.6: Success rates for separately distributed destinations.

4.3.2 End-to-End Delay

In this section we report simulation results for average end-to-end delay that is the time past 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. For

all scenarios, we can say that actually PBM performs worse than the reported results, but to calculate this difference we need parameters related to deployment and it is out of scope of this thesis.

Results of experiments for average end-to-end delay for randomly located destinations are shown in Figure 4.7. We observe that PBM and LBM-MST yield high average end-to-end delays than the LBM-D. We can also say that average 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 traverse the minimum spanning tree to reach the destinations which increases the end-to-end delays. Figure 4.7 also shows that average end-to-end delay for LBM-D decreases while the number of sensors increases.

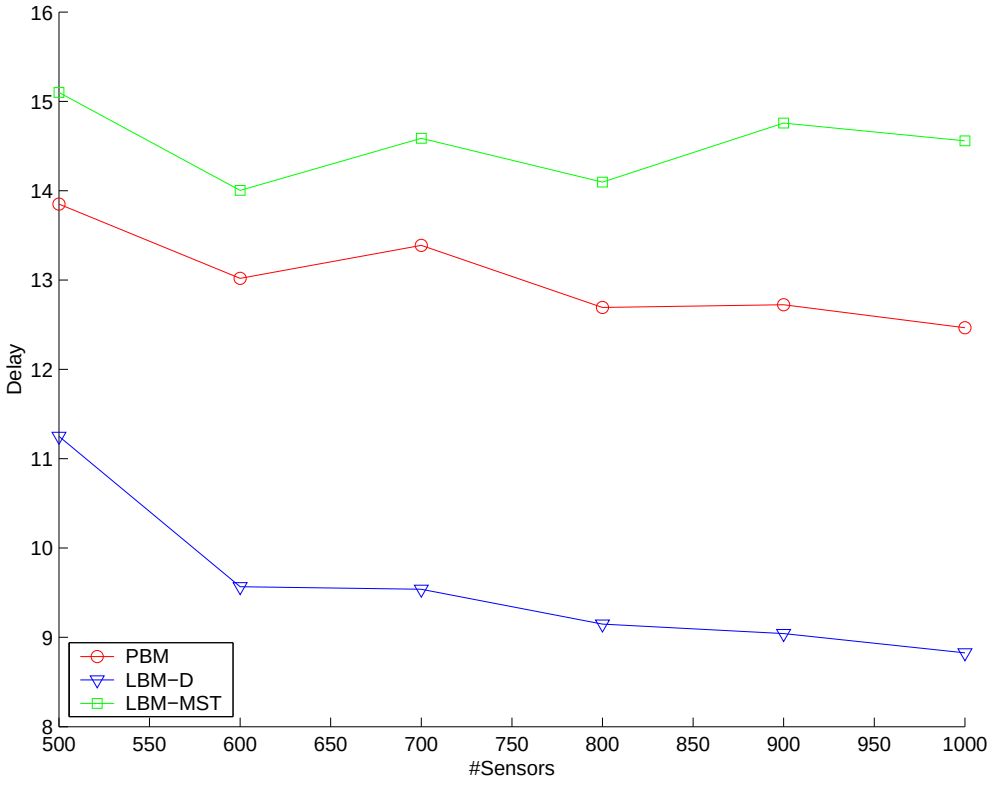


Figure 4.7: Average end-to-end delay for randomly located destinations.

Figure 4.8 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 performs better than LBM-MST for the same reasons mentioned before. For all three algorithms, average end-to-end delays decrease while the networks get denser.

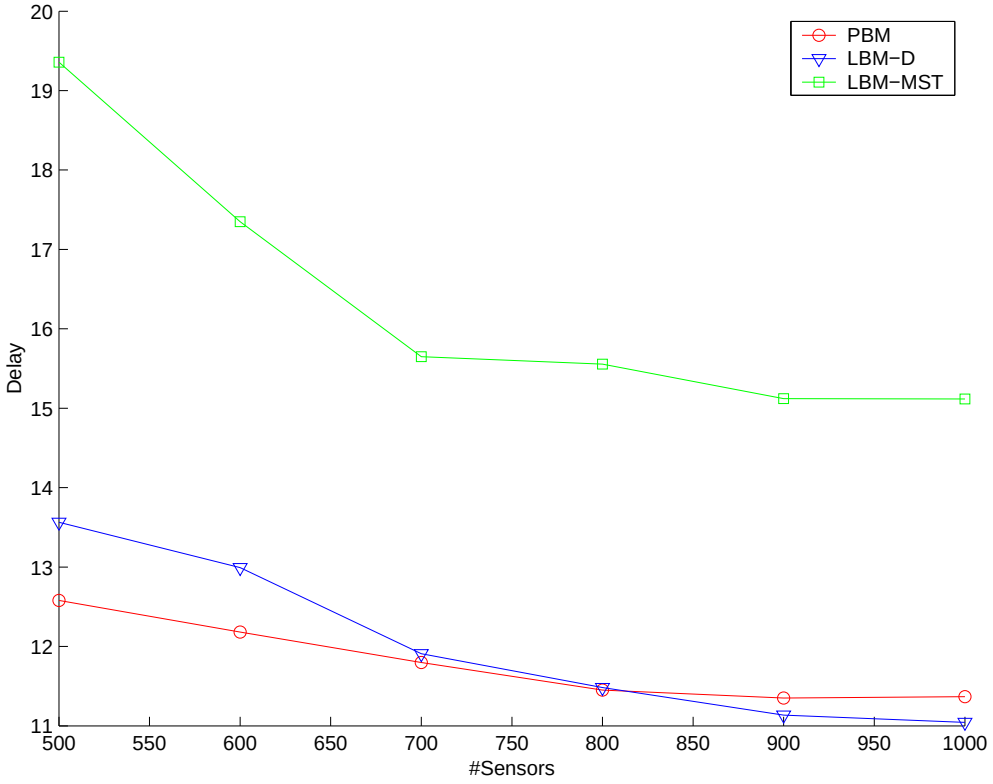


Figure 4.8: Average end-to-end delay for destinations grouped together.

In Figure 4.9, 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.

4.3.3 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

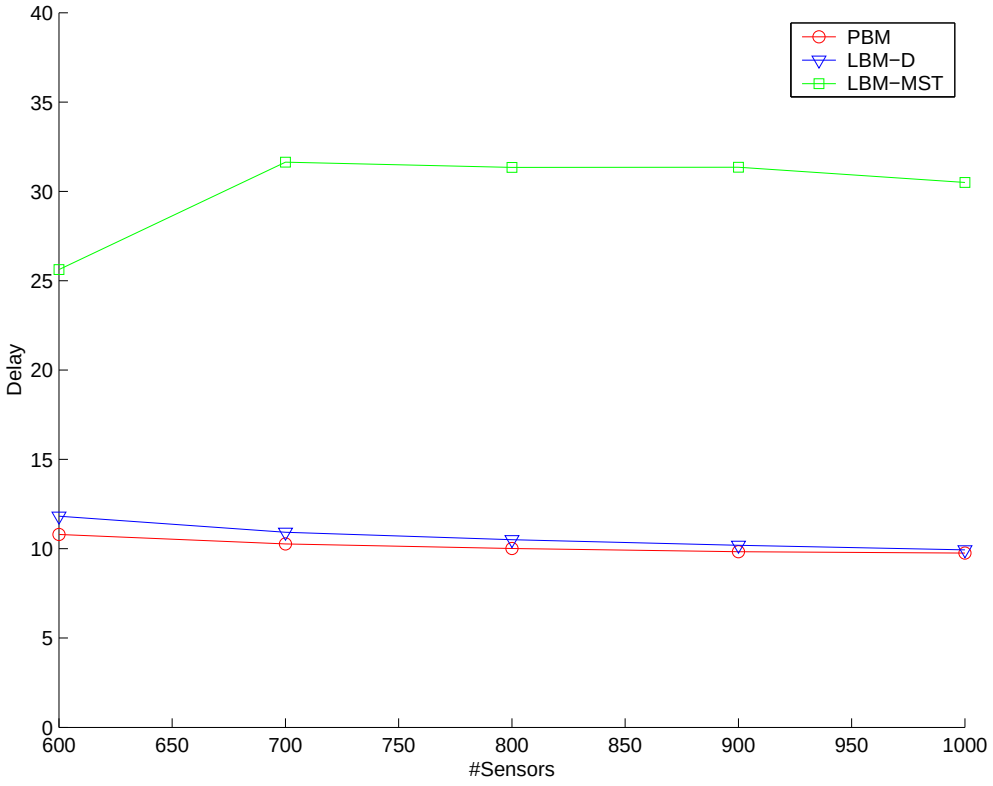


Figure 4.9: Average end-to-end delay for separately distributed destinations.

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. Therefor, 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 4.10 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 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 4.11. When all the destinations are located near 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 4.12. When the destinations are distributed to separated locations, LBM-MST uses less transmissions than PBM and LBM-D.

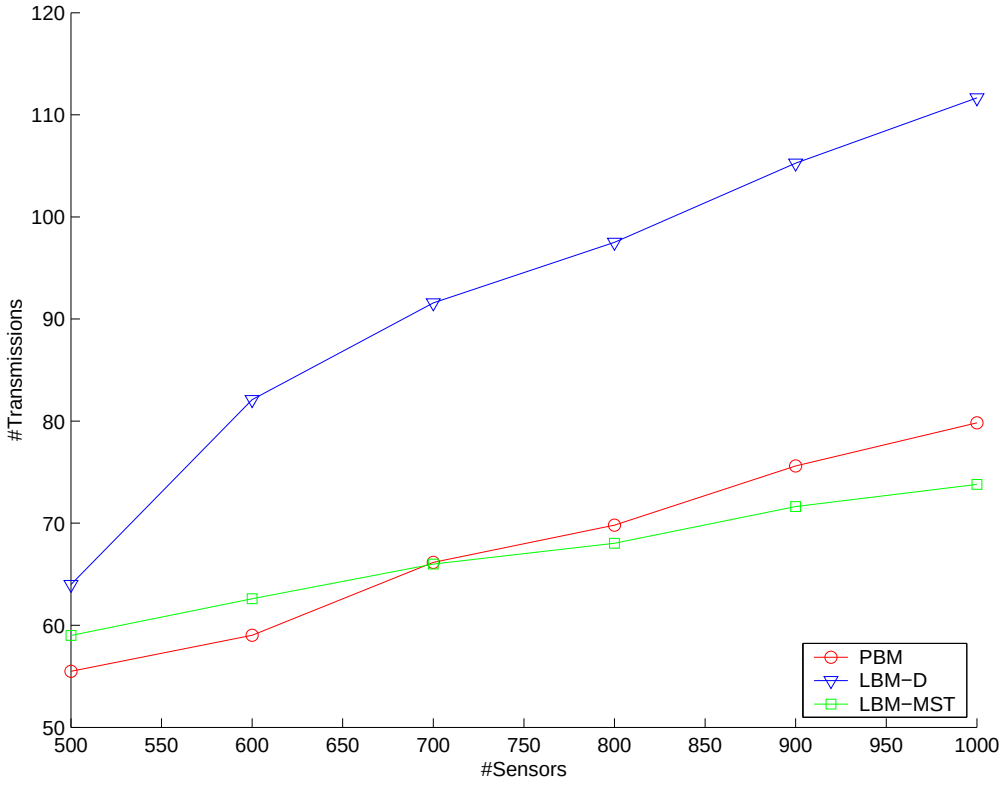


Figure 4.10: Number of transmissions for randomly located destinations.

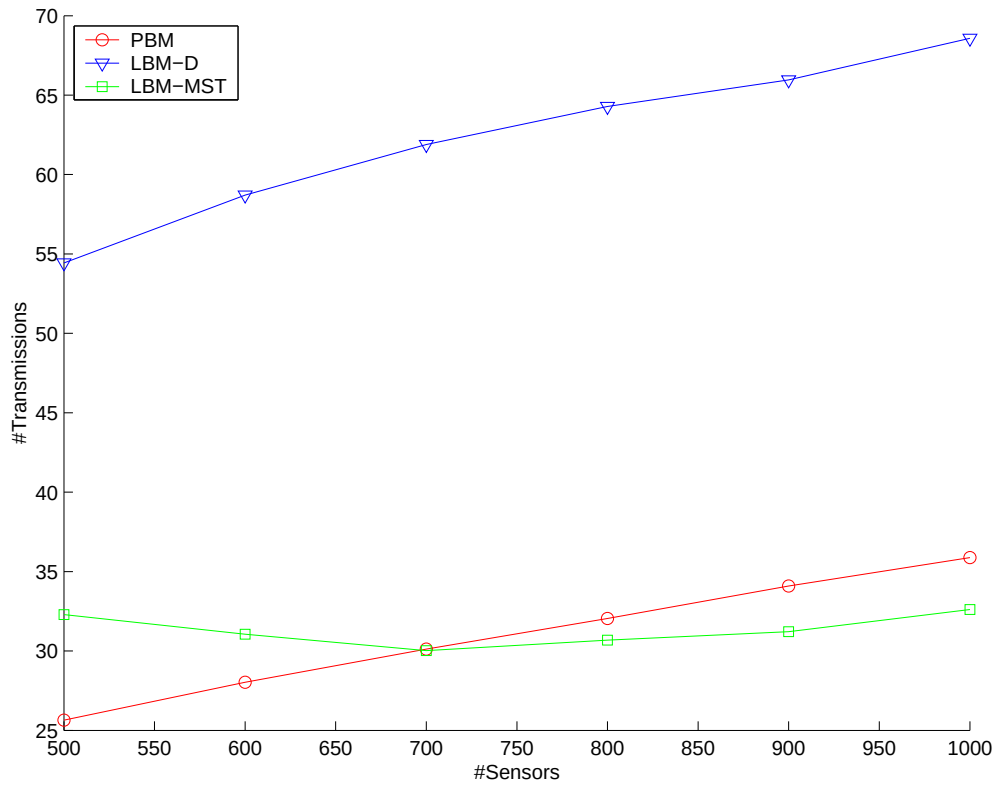


Figure 4.11: Number of transmissions for destinations grouped together.

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.

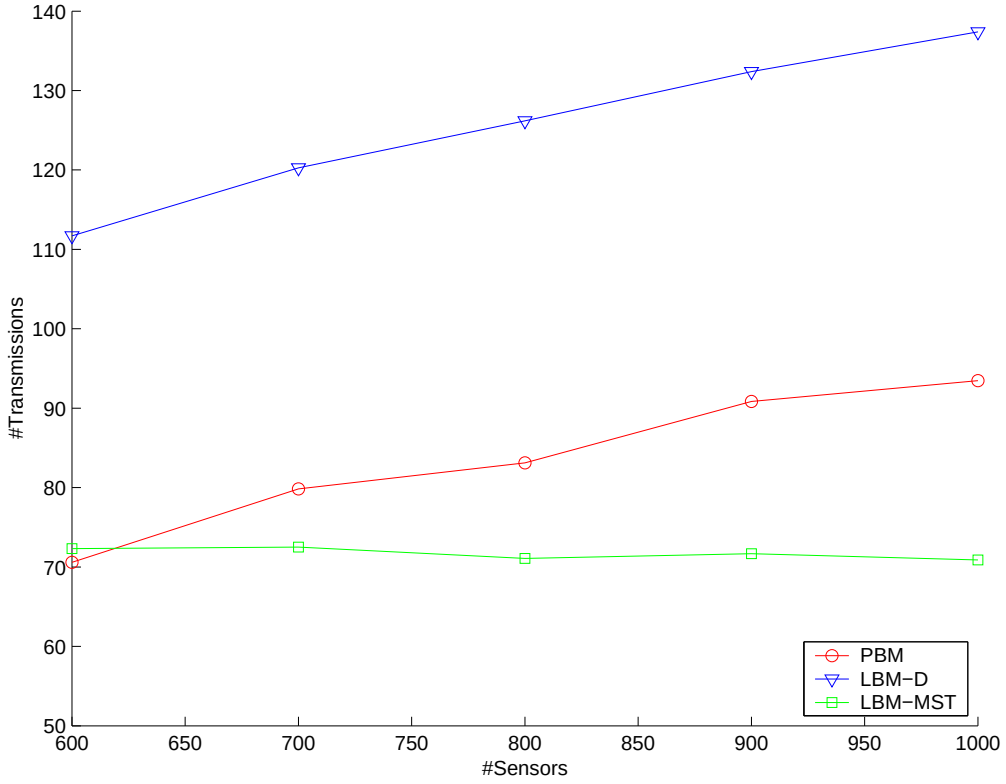


Figure 4.12: Number of transmissions for separately distributed destinations.

4.3.4 Traffic Overhead

In this section we evaluate the traffic overhead 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 this 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 for 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 4.13.

As seen in Figure 4.13, traffic overhead increases linearly while the actual data size grows for each algorithm. LBM-MST causes the minimum traffic overhead 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 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 4.14 depicts the result of this experiment.

Figure 4.14 shows that traffic overheads 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, traffic overhead 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 4.14. For example, when number of destinations are greater than 40, traffic overhead caused by LBM-MST is almost half of the overhead caused by other algorithms.

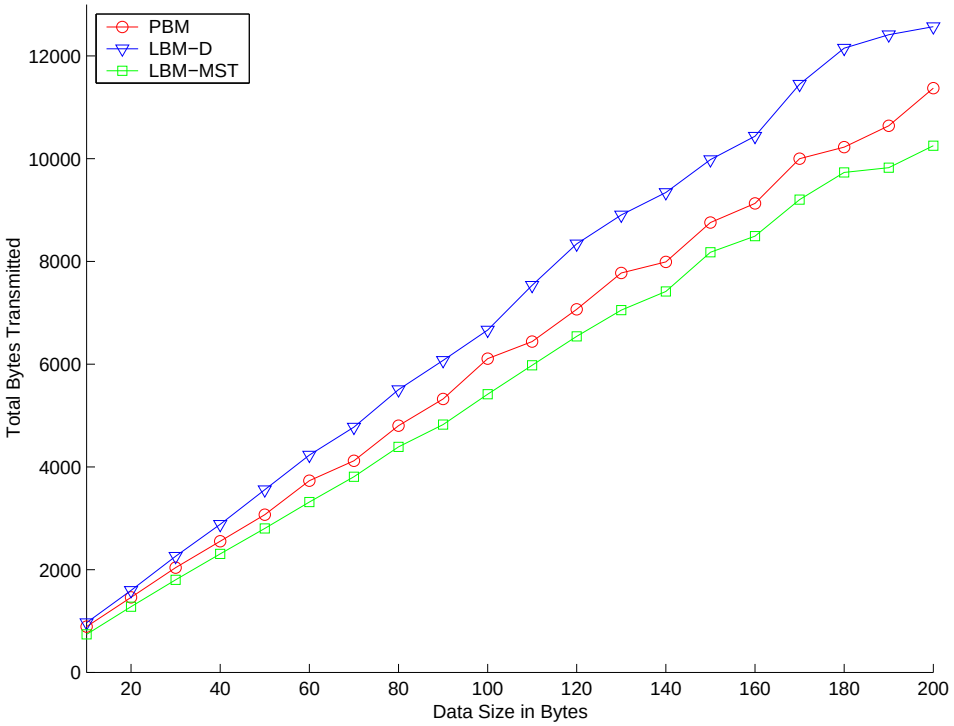


Figure 4.13: Traffic overhead vs. data size.

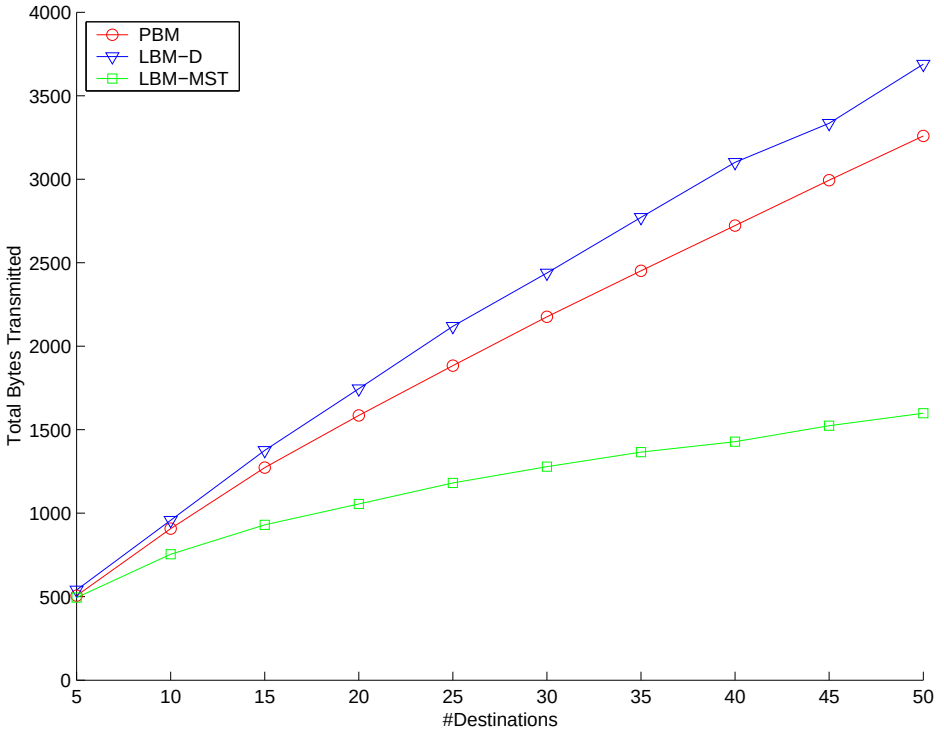


Figure 4.14: Traffic overhead vs. number of destinations.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis 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 our own simulator. We implemented the PBM [12] algorithm and our algorithms in this

simulation environment. We performed extensive tests to see if our algorithms work as intended. Then we evaluate our solutions by comparing them with PBM and also with each other.

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.

5.2 Future Work

The algorithms we propose can be improved with further study in many aspects. One of the essential concerns in sensor network algorithms is power consumption. Our algorithms tries to minimize the power consumption by decreasing the number of total transmissions. Beside this, they can be more power aware if we also consider the remaining energy levels of the sensor nodes while selecting the next neighbors to forward the multicast messages. This approach helps obtaining a power balanced network which has a longer lifetime.

The behavior of the algorithms can be also studied further. For instance, optimal angle values which are used to group the destinations can be searched for different positioning of the destinations. In addition, to increase success rate for LBM-MST, an adaptive approach can be developed which would try to route the message to other nodes of the MST when it cannot reach the next child.

LBM-D and LBM-MST both route the messages to multiple point locations

currently. They can be improved to support routing to multiple regions. This can be an interesting problem especially if the number of such regions are high. According to relative positioning of the regions, algorithms should find efficient paths to minimize the number of transmissions. Routing inside the regions can be accomplished with one of the existing geocasting protocols.

After having designed the algorithms, the next step might be implementing a complete location based multicast routing protocol for wireless sensor networks. This protocol can be evaluated in more sophisticated network simulators to see how practical it is. The last step of this study would be the deployment of the protocol on a real sensor network environment.

Bibliography

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[2] S. Basagni, I. Chlamtac, and V. Syrotiuk. Location aware, dependable multicast for mobile ad hoc networks. *Computer Networks*, 36(5-6):659–670, 2001.

[3] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks*, 7(6):609–616, 2001.

[4] D. Braginsky and D. Estrin. Rumor routing algorthim for sensor networks. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31, 2002.

[5] G. Finn. Routing and Addressing Problems in Large Metropolitan-scale Internetworks. *ISU-RR*, pages 87–180, 1987.

[6] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, 1999.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 56–67, 2000.

[8] R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 8(1):48–57, 2001.

[9] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. *Proc. 11th Canadian Conference on Computational Geometry*, pages 51–54, 1999.

[10] S. Lindsey and C. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. *Aerospace Conference Proceedings, IEEE*, 3:1125–1130, 2002.

[11] S. Lindsey, C. Raghavendra, and K. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):924–935, 2002.

[12] 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*, 7(3):53–55, 2003.

[13] C. S. R. Murthy and B. S. Manoj. *Ad Hoc Wireless Networks, Architectures and Protocols*. Prentice Hall, 2004.

[14] C. Perkins. *Ad Hoc Networking*. New York: Addison-Wesley, 2001.

[15] R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.

[16] I. Stojmenovic. Position-based routing in ad hoc networks. *Communications Magazine, IEEE*, 40(7):128–134, 2002.

[17] I. Stojmenovic and X. Lin. Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, 2001.

[18] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, 2002.

[19] H. Takagi and L. Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *Communications, IEEE Transactions on [legacy, pre-1988]*, 32(3):246–257, 1984.

[20] M. Transier, H. Fubler, J. Widmer, M. Mauve, and W. Effelsberg. Scalable Position-Based Multicast for Mobile Ad-hoc Networks. *Proc. of the First International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWim)*, 2004.

[21] C. H. Y. Y. Wentao Zhang, Xiaohua Jia. Energy-aware location-aided multicast routing in sensor networks. *Proc. International Conference on Wireless Communications, Networking and Mobile Computing*, 2:901–904, 2005.

[22] S. Wu and K. Candan. GMP: Distributed Geographic Multicast Routing in Wireless Sensor Networks. *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, 2006.

Appendix A

Table of Acronyms

MST	Minimum Spanning Tree
PBM	Position Based Multicasting
WSN	Wireless Sensor Network
SPIN	Sensor Protocols for Information via Negotiation
PEGASIS	Power-Efficient Gathering in Sensor Information Systems
CDMA	Code Division Multiple Access
LBM-D	Location Based Multicasting with Direction
LBM-MST	Location Based Multicasting according to Minimum Spanning Tree
GPS	Global Positioning System
MFR	The Most Forward within Radius
GEDIR	Geographic Distance Routing
DIR	Directional Routing
GRA	Geographic Routing Algorithm
SARF	Single Branch Regional Flooding
AP	Access Point
SAM	Single Branch Multicast Tree
CoFAM	Cone-base Forwarding Area Multicast Tree
MSAM	MST-based Single Branch Multicast Tree

SPBM	Scalable Position Based Multicasting
GMR	Geographic Multicast Routing
RepLoc	Representation of Location Information
SMS	Sent Message Signatures
GMP	Geographic Multicast routing Protocol
DSM	Dynamic Source Multicast