Journal of
*Sensor and Actuator Networks*

*Article*

# Distributed Algorithms for Multiple Path Backbone Discovery in Thick Linear Sensor Networks

**Imad Jawhar [1,*], Sheng Zhang [2], Jie Wu [3,*], Nader Mohamed [4] and Mohammad M. Masud [5]**

1 Faculty of Engineering, Al Maaref University, Beirut 1600, Lebanon
2 State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210000, China; sheng@nju.edu.cn
3 Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
4 Department of Computer Science, Information Systems and Engineering, California University of Pennsylvania, California, PA 15419, USA; mohamed@calu.edu
5 College of Information Technology, United Arab Emirates University, Al Ain P.O. Box 15551, United Arab Emirates; m.masud@uaeu.ac.ae
* Correspondence: imad.jawhar@mu.edu.lb (I.J.); jiewu@temple.edu (J.W.)

**Abstract:** Continued advancements in microprocessors, electronics, and communication technology have led to the design and development of sensing devices with increased functionalities, smaller sizes, larger processing, storage, and communication capabilities, and decreased cost. A large number of these sensor nodes are used in many environmental, infrastructure, commercial, and military monitoring applications. Due to the linearity of a good number of the monitored structures such as oil, gas, and water pipelines, borders, rivers, and roads, the wireless sensor networks (WSNs) that are used to monitor them have a linear topology. This type of WSN is called a linear sensor network (LSN). In this paper, two distributed algorithms for topology discovery in thick LSNs are presented: the linear backbone discovery algorithm (LBD) and the linear backbone discovery algorithm with x backbone paths (LBDx). Both of them try to construct a linear backbone for efficient routing in LSNs. However, the LBD algorithm has the objective of minimizing the number of messages used during the backbone discovery process. On the other hand, the LBDx algorithm focuses on reducing the number of hops of the data messages transmitted from the nodes to the sink. LBD and LBDx exhibit good properties and efficient performance, which are confirmed by extensive simulations.

**Keywords:** wireless sensor networks (WSNs); linear sensor networks (LSNs); routing; topology discovery

**Citation:** Jawhar, I.; Zhang, S.; Wu, J.; Mohamed, N.; Masud, M.M. Distributed Algorithms for Multiple Path Backbone Discovery in Thick Linear Sensor Networks. *J. Sens. Actuator Netw.* **2021**, *10*, 49. https://doi.org/10.3390/jsan10030049

Academic Editors: Lei Shu, Adnan Al-Anbuky, Stefan Fischer, Joel J. P. C. Rodrigues and Mário Alves

Received: 8 May 2021
Accepted: 29 June 2021
Published: 16 July 2021

## 1. Introduction

Many structures or regions that need to be monitored by wireless sensor networks (WSNs), such as oil, gas, and water pipelines, rivers, borders, roads, and coast lines, exhibit a linear form. Consequently, the resulting topology of the WSN leads to the creation of a linear sensor network (LSN) [1]. The networking and routing protocols that are used to transmit the data in LSNs can be designed to take advantage of the linearity of the network to improve various aspects, including performance, reliability, energy consumption, and routing efficiency.

As was mentioned earlier, there are many applications for LSNs. In some of these applications, the sensor nodes (SNs) are deployed by throwing them in a semi-random form from an unmanned aerial vehicle (UAV) along a thick formation between two parallel lines [2]. The resulting network of sensors constitutes a thick LSN [1], which is illustrated in Figure 1. Such thick LSN applications include border monitoring of territories that are uninhabitable or unreachable by human installation and service personnel due to natural, political, or military reasons. They also include sea coast and river monitoring, monitoring the areas around natural or manmade linear structures such as oil, gas, and water pipelines, railroads, and subways. For an LSN that might extend to tens or hundreds of kilometers, the network can be divided into multiple segments that are separated by sink nodes. Such sink nodes can also be deployed by throwing them from low-flying UAVs with an average distance separating them [3]. Another option is to install the sink nodes with network personnel if the terrain is easily reachable.

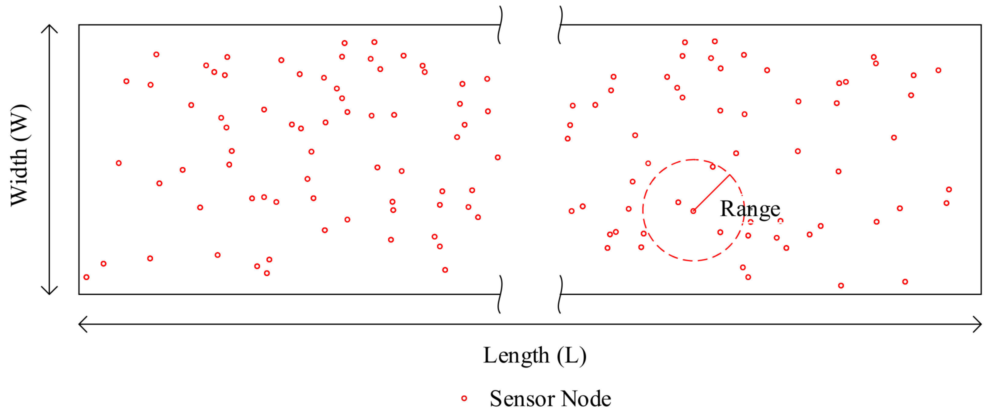


**Figure 1.** Illustration of a thick linear sensor network. Sample parameter values: L = 10,000 m, W = 500 m, and range = 100 m.

In this paper, we consider the thick LSN scenario. For such a network, we want to design a good topology discovery algorithm to improve routing efficiency and reliability. To this aim, we propose two graph search-based topology discovery algorithms, which are the linear backbone discovery algorithm (LBD) and linear backbone discovery algorithm with x backbone paths (LBDx). LBD tries to construct a backbone path from the source node to the sink node and uses this backbone path to relay messages. Different from LBD, LBDx constructs x backbone paths for message relaying, where x is a flexible parameter and can be adjusted by the operator. Both of them have their respective advantages. Since LBD only constructs one backbone path, it incurs less construction messages than LBDx; however, LBDx has more backbone paths than LBD, making non-backbone nodes in LBDx have less communication hops from the source or sink than in LBD. We will see shortly in later sections that x in LBDx can be used to tradeoff between reducing construction messages and minimizing the average communication hops. Overall, LBD and LBDx have different objectives: LBD concentrates on minimizing the number of backbone construction messages [4], while LBDx can flexibly control the tradeoff between construction overhead and communication length. Consequently, the algorithms result in the selection of certain SNs to form a backbone that can be used to transmit messages from all of the SNs to the sink nodes at the end of the LSN or LSN segment.

The discovery of the backbone provides several advantages over many of the other routing protocols that have been proposed for multi-dimensional WSNs. It provides scalability, since only one backbone consisting of one or more paths is needed to be used for the transmission of messages for all of the SNs in the network. The number of these SNs might be as large as hundreds or thousands of nodes. It also results in increased reliability by allowing the SNs to increase their transmission range to jump over failed nodes due to the a priori knowledge of the linearity of the network. Another option that can be used to enhance routing reliability is to propagate the message in the opposite direction to reach the sink at the other end of the network [1]. Furthermore, the SNs do not need to be equipped with GPS capabilities, which can significantly increase their cost and energy consumption. In order to verify the operation of the protocols and evaluate their performance, simulation experiments were conducted under various network conditions. Consequently, this paper has the following contributions:

- Offer new backbone discovery algorithms for thick linear sensor networks. The algorithms take advantage of the linearity of the topology in order to increase their efficiency and lower the overhead of the exchanged control messages.
- The discovered backbone can be used for the transmission of data messages to the nearest sink, which is located at the end of the LSN.
- We studied the performance of the offered algorithms under various network conditions, which allows for the proper selection of the appropriate algorithm, depending on the network characteristics.

The rest of the paper is organized as follows. Section 2 offers related work. Section 3 presents the operation of the LBD algorithm. Section 4 provides an extension of LBD and discusses LBDx. Section 5 offers the simulation experiments, results, and related discussions. Section 6 concludes the paper.

## 2. Related Work

Some research has been done to address the various issues in LSNs or one-dimensional (1-D) networks. Wireless capacity with 1- D mobility was studied in [5]. An approximation formula for the connectivity probability for 1-D ad hoc networks was derived in [6]. A queueing theory approach was used to study the same problem in [7].

Topology discovery techniques for ad hoc networks have been studied by many researchers. These techniques can be classified into four classes [8,9]. One technique is based on node discovery [10–12]. In [10], Singular value decomposition was used to obtain the topology map from the virtual coordinate system. In [11], sampling of the node IDs was adopted to derive the node estimation and topology discovery. In [12], random walk and iterative multi-lateration localization strategies were used to implement physical topology discovery.

Another technique uses energy conservation though sleep scheduling in the process of topology discovery. For example, in [13], a selective node wake-up schedule along with collision avoidance was used to control topology discovery. A third technique considers power control in the topology discovery process. In [14], controlling the transmission power of the node was done in order to achieve efficient topology discovery. In [15], modeling of the topology discovery was done under the constraint of saving node energy while preserving graph connectivity. In the fourth technique, a hierarchical approach was used. In [16], a set of nodes that was able to act as cluster heads was selected. In [17], the total transmission power of the nodes was minimized by using the selected resource-rich nodes as cluster heads. The other simple nodes with limited energy capacity used these nodes for their transmission process.

In [18], the authors proposed distributed algorithms for finding a dominating set-based virtual backbone for routing in asymmetric (i.e., nodes have a non-uniform or varying transmission range) MANETs. In [19], a quality-of-service, load-balanced, connected dominating set backbone discovery algorithm for MANETs was proposed. In [20], distributed dynamic backbone-based flooding in unstructured networks was presented. In [21], the authors presented an algorithm for backbone routing in hierarchical MANETs. Another backbone discovery algorithm using a minimum spanning tree in cognitive radio networks is shown in [22]. In [23], a backbone was constructed for wireless visual sensor networks (WVSNs) where the nodes used a directional antenna. The constructed backbone was intended to be used for data aggregation. In [24], a centralized algorithm to build a virtual backbone in a WSN is presented. This is done through finding the core and supporting sensors using the dominating set strategy. The core nodes dominate the supporting nodes, and the latter ones dominate the rest of the nodes in the WSN. In [25], a backbone was constructed in multilayer ad hoc networks using a connected dominating set (CDS). The CDS was extended with nodes that were highly connected and energy-rich. In [26], UAVs acting as relay nodes were used to construct a backbone to connect with terrestrial ad hoc networks in disaster scenarios. In [12], the authors proposed a topology discovery algorithm for a WSN using a random walk strategy.

Other strategies adopt fault tolerance as a goal for their topology discovery algorithms [27,28]. For example, in [28], k-edge connectivity with an approximation factor was maintained while minimizing power. In [12], an overview of the topology algorithms that have been proposed in the research for multi-dimensional WSNs is provided. In addition, a survey of topology management and control algorithms is offered in [29].

Researchers have proposed routing algorithms for mobile ad hoc networks (MANETs) and WSNs [30]. Several of these algorithms construct a backbone for WSNs with a mobile sink. The backbone strategy is used to increase routing efficiency, especially with networks containing a large number of nodes. In [31], the protocol used starfish routing with a mobile sink, which built a backbone based on the network size and node transmission range. In [32], a minimal spanning tree construction method was used along with the creation of a connected dominated set. In [33], scheduling node transmissions was performed by constructing and using multiple overlapped backbones. Forming a fully connected backbone with a minimal number of nodes was demonstrated as an NP complete problem. In [34], three randomized algorithms to form connected dominating sets are presented. In [35], the ant colony optimization (ACO) strategy was used to construct an energy-efficient connected dominating set. The algorithm was evaluated in comparison with the genetic algorithm (GA) backbone construction strategy. In [36], two-phase geographic greedy forwarding (TPGF) routing algorithm for wireless multimedia sensor networks (WMSNs) is presented. The algorithm supports hole bypassing, shortest path transmission, and multipath transmission. In addition, an energy-aware geographic routing protocol with sleep scheduling for WMSNs is provided in [37]. A survey of multipath routing in WSNs is offered in [38], which includes a comparison of these algorithms.

Our approach differs from these strategies in several ways and, as a result, has the following advantages:

- The above algorithms are proposed for multi-dimensional WSNs. They do not take advantage of the a priori knowledge of the linearity of the network.
- The algorithms presented in this paper are designed to discover a backbone for LSNs, which is later used for routing. The discovery process is designed to keep the number of overhead messages low and increase communication efficiency, scalability, reliability, and fault tolerance.
- Our technique is more general and takes advantage of the linearity of the network topology. It does not use the energy of the nodes as a constraint and does not use a hierarchical model, which leads to depletion of the energy of the nodes which are used as cluster heads.
- Our algorithms have the ability to provide load balancing through the use of multiple paths, which distributes the energy consumption among the nodes that constitute the paths of the backbone (in the case of LBDx).
- Due to the prior knowledge of the linear nature of the backbone, localized path repair to go around failed nodes can be done in a much easier fashion, since the direction of the propagation of the messages is known.
- Our algorithms are not based on power control, as is the case in some of the algorithms listed above, which requires more complex and expensive circuitry. Our algorithms use simpler and less expensive sensor nodes.
- Unlike some of the algorithms listed above, our algorithms are not based on sleep scheduling, which increases the algorithm's end-to-end delay for backbone discovery and subsequent data transmissions.

## 3. Linear Backbone Discovery (LBD)

LBD tries to construct a backbone path from the source node to the sink node using the smallest number of construction messages. The backbone path can be used to forward messages and thus improve routing efficiency. LBD contains two phases: backbone discovery (BD) and new backbone node declaration (NBND). In this section, we first introduce the BD algorithm and then explain the NBND algorithm.

### *3.1. Phase 1: Backbone Discovery (BD)*

The backbone discovery (BD) algorithm contains three components: BD at the source node (Algorithm 1), BD at an intermediate node (Algorithms 2 and 4), and BD at the sink node (Algorithm 3). The main notations used in this phase are summarized in Table 1 for quick reference. Figure 2 shows a running example of BD, the details of which are described subsequently.

**Table 1.** Main notations used in the BD phase.

| Symbol | Meaning |
|---|---|
| messageID | The ID of a message |
| messageLc | The number of nodes in the discovered path from the source node |
| myID | The ID of a node |
| myLc | The location of a node from the source node |
| PATH | An ordered list of nodes in the discovered path from the source node |
| myParent | The last node before a node in PATH |
| myBNeigh | The backward neighbor of a node in the backbone path |
| myFNeigh | The forward neighbor of a node in the backbone path |
| BBLc | The location of a node in the backbone path from the source node |
| BBLcFromSink | The location of a node in the backbone path from the sink node |
| BB | A Boolean variable indicating whether a node belongs to the backbone path |

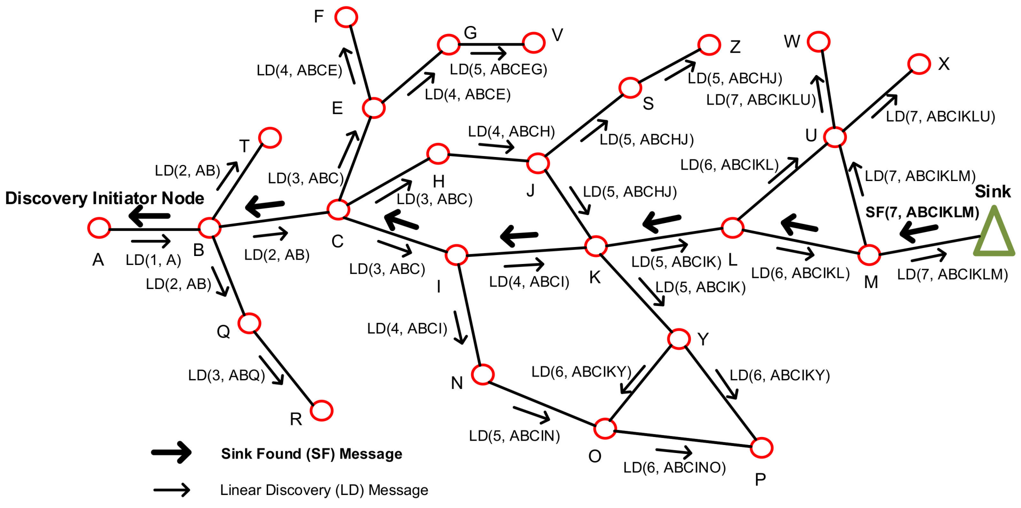


**Figure 2.** Propagation of an LD message in the LBD algorithm [39].

*(1) BD at the Source Node:* We will first introduce a few necessary notations: messageID contains the identification of a message, messageLc is the number of nodes in the discovered path from the source node, myID is the identification of a node, and myLc for each node is initiated as follows. If it is the source node, its myLc is set to 0; otherwise, it is set to 1. The variable PATH contains an ordered list of the nodes that constitute the discovered path, while myParent stores the last node of a node in the discovered path from the source node, and BB is a Boolean variable indicating whether a node belongs to the backbone path.

Algorithm 1 describes the steps taken by the source node at the primary edge of the network. The node initializes myParent to NULL and the backbone flag BB to TRUE, since this node is already included in the backbone. It also includes its ID (myID) in the PATH list and broadcasts the constructed linear discovery message LD(messageID, myID, messageLc, PATH) to all of its neighbors. This starts the backbone discovery process.

*(2) BD at an Intermediate Node:* Algorithm 2 shows the actions taken by a node y upon receiving the LD message from a node x. First, node y checks to see if the message's messageLc counter is less than (i.e., better) than its myLc counter. If that is the case, then it sets x to be its new parent node, updates its counter with that of the message, increases the

messageLc counter by 1, adds its ID to the message, and broadcasts the message to all of its neighbors; otherwise (i.e., the messageLc counter is not less than its message counter myLC), it drops the message.

*(3) Message Propagation Delay Issues:* It is important to note here that it is possible to have discovery message delays at certain nodes due to congestion problems or other delivery latency issues. This may lead to the arrival of the LD message with a longer accumulated PATH to an intermediate node first. For example, in Figure 2, node K might receive an LD message from node J with the longer PATH ABCHJ before receiving it from node I with the shorter PATH ABCI. In our algorithm, the intermediate node will eventually update its table and forward an LD message with the shorter PATH, due to the fact that it compares its myLc counter with that of the message messageLC and forwards the LD message with the smaller path if the messageLC counter is smaller.

**Algorithm 1:** BD at the source node

```
/* Broadcasting the LD message from the source node */
myParent = NULL
BB = TRUE
if (this is the first node at the primary edge) then
myLc = 0
    messageLc = 1
    PATH = myID
    SendLD(messageID, myID, messageLc, PATH) to all neighbors
else
    myLc = $\infty$
end if
```

**Algorithm 2**: BD at an intermediate node. Reception of an LD message.

```
/* The LD message, LD(messageID, x, messageLc), arrives at node y from node x. */
myParent = NULL
if (myLc $\geq$ messageLc) then
    myParent = x
    myLc = messageLc
    messageLc++
    Concatenate myID at the end of the PATH list
    Broadcast LD(messageID, myID, messageLc, PATH) to all neighbors.
else
    Drop LD message
end if
```

*(4) BD at the Sink Node:* When the sink receives the LD message from a node x, it executes the steps outlined in Algorithm 3. Specifically, it sets the BB flag to TRUE, indicating that it is a part of the backbone. It sets it parent, myParent, and backward neighbor, myBNeigh, to x. It sets its forward neighbor, myFNeigh, to NULL. It sets its backbone linear count counter, BBLc, to the linear counter in the message, messageLc. It sets the backbone linear count from the sink counter BBLcFromSink to 0. It then unicasts a *Sink Found* message, SF, back along the discovered backbone with the node IDs accumulated in PATH back to the source.

**Algorithm 3:** BD at the sink node

```
BB = TRUE/
*When the sink receives the LD(messageID, x, messageLc, PATH) message from node x it does the following:*/
myPrarent = x
myBNeigh = x
myFNeigh = NULL
BBLc = messageLc
BBLcFromSink = 0
Send SF(messageID, source = myID, destination = x, BBlc, PATH)
```

*(5) BD at an Intermediate Node (receiving an SF message from the sink:* When an intermediate node y receives an SF message from another node x, it takes the steps described in Algorithm 4. Specifically, it sets the backbone flag BB to TRUE, indicating that it is a part of the discovered backbone. Then, it caches the information (ordered list of backbone node IDs) of the discovered backbone stored in PATH. It can either cache the full path (IDs of all of the nodes in PATH) or only a local part (k nodes forward and k nodes backward). In the former strategy, the node uses more memory. However, it has full flexibility in forwarding data in either direction in the future. If one of the paths (or directions) to reach a sink fails, the node can then send the data message in the opposite direction. This provides more flexibility and fault tolerance. On the other hand, if the latter strategy is applied, less memory is used by the node. However, it does not have the option to use source routing to forward the packet all the way to the sink in the opposite direction, if that is needed, due to path failure in the first direction. After this information caching is done, the node then sets its backward neighbor myBNeigh as its parent, its forward neighbor myFNeigh to x, its backbone linear counter BBLcFromSink to myLc, and its backbone counter from the sink BBLcFromSink to messageLc—myLc. Finally, it propagates the SF message to the next node down the line of the discovered backbone stored in PATH. Consequently, each subsequent node similarly updates its information and propagates the SF message further until the latter reaches the source node, which completes the backbone discovery process. At the conclusion of the BD phase, the nodes in the network would be classified into two types: *backbone nodes (BNs)*, which constitute the discovered backbone, and *non-backbone nodes (NBs)*, which are the rest of the nodes in the network. The NB nodes have sensing responsibilities and can forward their data to the nearest BN node using the algorithms that are described in the next section. In the later stages, if the nodes in the current backbone are depleted and need to be changed, subsequent discovery processes might cause some of these NBs to become new BNs. This strategy is being considered for future research.

**Algorithm 4:** BD at an intermediate node. Reception of an SF message from the sink.

```
/*When node y receives the SF(messageID, x, messageLc, PATH) message from node x it does the following:*/
BB = TRUE
Save the full or local part of the discovered backbone in PATH in the routing table according to the adopted caching strategy.
myBNeigh = myParent
myFNeigh = x
BBLc = myLc
BBLcFromSink = messageLc − myLc
Send SF(messageID, source = myID, destination = x, messageLc, PATH)
```

## 3.2. Phase 2: New Backbone Node Declaration (NBND)

Phase 1 (i.e., the BD phase) finds a backbone path from the source node to the sink nodes. Each node in the network knows whether it is in the backbone path, as each node maintains a Boolean variable BB. However, the non-backbone nodes (i.e., NBs) do not know which nodes are backbone nodes. In this phase, we introduce the new backbone node declaration (NBND) algorithm for each backbone node to "broadcast" its role information. Here, by "broadcast" we mean a small scale and well-controlled broadcast, which will be clear shortly. NBND contains two algorithms (shown in Algorithms 5 and 6). Algorithm 5 gives the NBND at the BN node, while Algorithm 6 gives the NBND at the NB nodes. Figure 3 illustrates the propagation of the NBD message in the NBND algorithm. The main notations used in this phase are summarized in Table 2 for quick reference.

**Algorithm 5:** NBND at the backbone nodes

```
    sourceBNID = myID
    NBDringSize = ρ
    numOfHops = 0
    PATH_TO_BN = myID
    Broadcast NBD (messageID, sourceBNID, myID, NBDringSize, numOfHops, PATH_TO_BN)
```

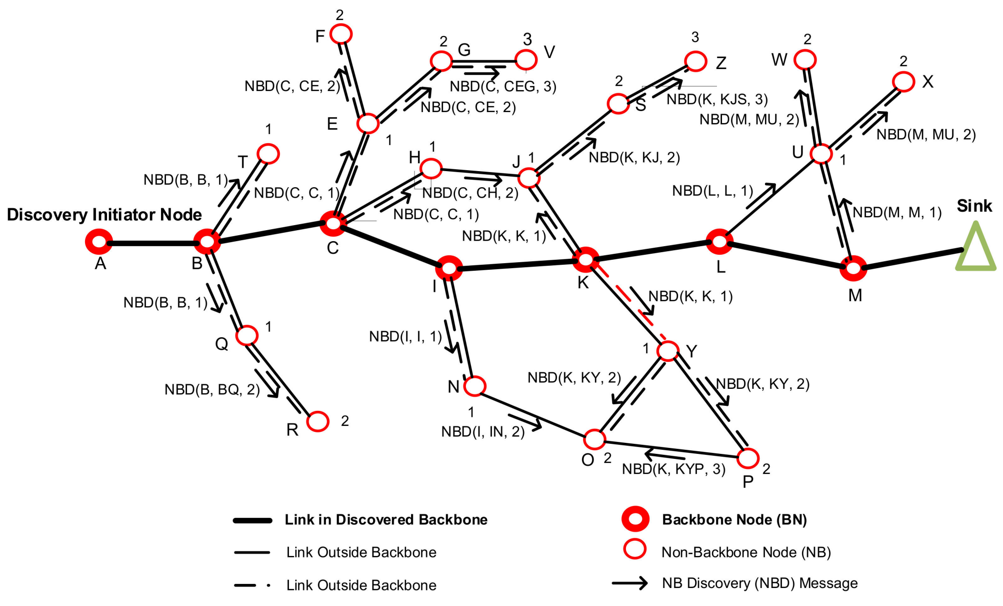


**Figure 3.** Propagation of the NBD message in the NBND algorithm [39].

**Table 2.** Main notations used in the NBND phase.

| Symbol | Meaning |
|---|---|
| sourceBNID | The myID of the sending BN node |
| BNDringSize | Set to ρ, which is the broadcast ring size in a number of hops |
| numOfHops | Cumulative number of hops traversed by the message |
| PATH_to_BN | Ordered list of nodes to reach the newly discovered BN in the backbone |

*(1) NBND at the backbone node:* In order to know what the nearest backbone node from each non-backbone node is, the NBND at the backbone node should "broadcast" its role information to all the other nodes. However, if we truly broadcast such information, there will be too many messages flooding the network. Therefore, we only broadcast the role information of each backbone node within ρ hops from each node.

When a node becomes a part of the backbone (i.e., becomes a BN node), it executes the NBND algorithm to allow the surrounding NB nodes to discover a path to itself. This enables the NB nodes to send their collected data to the sink through the nearest BN node to reach the backbone and the sink, respectively. This process is described in Algorithm 5. It works in the following manner. The source BN node constructs the new backbone node message NBD. The message includes the following parameters: messageID, which is the ID of the message and is used to prevent looping; sourceBNID, which is the ID of the BN node that initiated the message; myID, which is the ID of the node that is sending the message; NBDringSize, which is the size of the NBD message propagation ring and is used to keep the message from flooding the entire network and increasing the discovery overhead; numOfHops, which holds the number of hops that the message has traversed; and PATH TO BN, which has an ordered list of the accumulated IDs of the nodes along the constructed path to the BN node that initiated the NBD message.

*(2) NBND at the non-backbone node:* When an intermediate node y receives the NBD (messageID, sourceBNID, myID, BNDringSize, numOfHops, PATH TO BN) message from another node x, it performs the steps outlined in Algorithm 6, which are the following. It caches the accumulated path to the source BN node PATH TO BN. Then, it increments the number of hops to the BN node numOfHops by 1. Then, it checks to see if the new number of hops is still less than or equal to the NBD message ring size, NBDringSize. If that is the case, then it adds its ID, myID, to the accumulated path, PATH TO BN, and broadcasts the updated NBD message to all of its neighbors. Otherwise, numOfHops is greater than NBDringSize, and the ring size of the NBD message is exceeded. Therefore, node y drops the message.

The propagation of the NBD message is shown in Figure 3. In the figure, the nodes that belong to the discovered backbone are A, B, C, I, K, L, and M. Each of these BN nodes broadcasts an NBD message, which has the parameters specified earlier. The figure shows the message with the source node ID and the constructed path to the BN node, along with the number of hops it takes the NB node to reach the corresponding BN node in the backbone. For example, node C sends the NBD message, which is received by nodes E and H with one-hop paths, which are CE and CH, respectively. Node K also sends the NBD message, which is received by nodes J and Y with one-hop paths, which are KJ and KY, respectively. Node P receives the NBD message sent from node K with a two-hop path, which is KY P. Additionally, node Z receives the NBD message sent from node K with a three-hop path, KJSZ. As the NBD messages propagate, each NB node receives one or more NBD messages from different BN nodes, which are within the NBD message propagation ring hop limit. However, each NB node only caches the shortest path to the nearest BN node. Consequently, the NB node is able to use this path to send data to the sink through the backbone using the nearest BN node.

**Algorithm 6:** NBND at the non-backbone nodes

```
The message NBD (messageID, sourceBNID, myID, BNDringSize, numOfHops, PATH_TO_BN)
is received by node y from x.
Cache PATH_TO_BN, which constitutes the path from y to the BN node that sent the NBD
message in the routing table.
numOfHops++
if (NBDringSize $\geq$ numOfHops) then
    Concatenate myID at the end of the PATH_TO_BN list
    Broadcast the message NBD (messageID, sourceBNID, myID, BNDringSize, numOfHops,
PATH_TO_BN)
else
    Drop the received NBD message
end if
```

## 4. Linear Backbone Discovery with x Backbone Paths (LBDx)

In this section, we first motivate the design of LBDx and then present the details of LBDx.

### *4.1. Motivation*

As was discussed earlier, the LBD algorithm discovers a backbone that consists of the shortest path from the source to the sink. At the end of the discovery process, the nodes in the network are classified into BN nodes, which are a part of the backbone, and NB nodes, which are not a part of it. Using the NBND algorithm, each one of the NB nodes discovers the shortest path from itself to the nearest BN node. When the NB node needs to transmit data to the sink, it uses this path to reach the BN node. Then, the latter transmits the data to the sink through the rest of the BN nodes in the backbone and in the direction of the sink. In the design of such a routing algorithm for thick LSNs, two parameters that affect its efficiency can be considered as dimensions (of the design space). These parameters are the number of control messages exchanged during the backbone discovery process and the average number of hops to send data messages from the NB nodes to the sink using the constructed backbone.

It is reasonable to say that the LBD algorithm reduces the number of construction messages, since each node in the network does not have to discover its own shortest path to the sink. However, the average number of communication hops from each node to the sink is not minimal. We would like to see if a trade-off between the two parameters is possible. Before we come up with an algorithm that performs such a tradeoff, we consider a baseline naive algorithm, which we call S*, that allows each node to discover the shortest path from itself to the sink by flooding the network with its own LD message. Consequently, the S* algorithm results in a small average number of communication hops, since each node uses its own shortest path to the sink. However, it is not hard to see that the S* algorithm results in an unacceptable amount of construction messages and discovery overhead. Clearly, the LBD and S* algorithms represent two extremes for the routing process from the nodes to the sink in a thick LSN. The LBD algorithm has the objective of reducing the number of construction messages, while the S* algorithm does the opposite.

We thereby propose a general framework for trading off between the construction overhead and communication length. We call this framework LBDx, where x denotes the number of backbone paths between the source and the sink nodes and can also be considered as another important dimension of the design space of the backbone discovery process. LBDx constructs x backbones between the source and sink nodes. The objective is to strike a balance between the two extremes stated earlier.

### *4.2. Design Details*

In Algorithm 7, we use "I" and "S" to denote the initiator and sink nodes. The algorithm requires x anchor nodes, which are labeled as Ai. In WSNs, it is normal to designate some nodes as anchor nodes. Such nodes might be used for localization or other purposes. Furthermore, the position of the anchor nodes can be flexibly adjusted as needed by the WSN. It is desirable to have these anchor nodes in the mid-range along the length of the LSN (i.e., horizontally) between the initiator and sink nodes and equally spaced from top to bottom along the width of the LSN (i.e., vertically).

In the algorithm, we use the LBD algorithm to discover a backbone from node I to node Ai and another one from node Ai to node S. This is done for each i as i goes from 1 and x. Then, we join the backbones I-Ai and Ai-S to construct x backbones from node I to the sink node S. Afterward, the NBND algorithm is used by the discovered backbone nodes to allow the NB nodes to discover paths from themselves to the newly discovered BN nodes in the backbones. The LBDx algorithm is flexible, allowing the choice of the number of anchor nodes to vary depending on the width (or length) or height (or thickness) of the LSN, as well as the node density. Although the number of generated LBD messages increases with the number of paths and anchor nodes, compared with the LBD algorithm, the average number of communication hops from the nodes to the sink is expected to decrease.

In order to verify the operation of LBDx and study its performance, we will take a closer look at LBD2, where x = 2. The latter algorithm uses two anchor nodes A1 and A2. The location of these two nodes can be adjusted according to the parameters of the thick LSN. Let us consider the LSN to be in a rectangular area with a length L and thickness T, where the top left corner point has the coordinates of (0, 0). Then, anchor node A1 can be placed at location (L/2, T/4), and anchor node A2 can be placed at location (L/2, 3T/4). The operation of LBD2 is shown in Algorithm 8.

### *4.3. Mitigating Node Failures*

Due to the limited battery capacity of the sensor nodes in WSNs, energy consumption is a major concern. Consequently, in order to mitigate node failures and increase the network's lifetime, several strategies can be employed in LBDx:

- Increasing x leads to an increased number of paths to reach the sink. Therefore, the load for forwarding data messages from the SNs is spread across a larger number of backbone paths and backbone nodes. This leads to lower energy consumption in the backbone nodes and increases their lifetime.
- Periodically reinitiating backbone discovery to discover new backbone paths by changing of the anchor nodes. Different strategies can be considered in order to achieve this objective.
- Periodic changing of the value of x and reinitiating backbone discovery can be performed in order to generate new backbone paths with different backbone nodes. This allows for more spreading of the forwarding load and leads to a lower rate of failure of the backbone nodes.

Since backbone sensors have limited battery capacity, they are subject to battery depletion due to message forwarding. The detection of node failure can be conducted using various strategies, including the employment of the following:

- Periodic hello messages, which are regularly transmitted by the node;
- Hop-to-hop acknowledgements, which are transmitted when messages are forwarded from one node to another;
- Passive acknowledgement, which consists of a node listening for and confirming the forwarding of the message by its neighbor.

Once failed nodes are detected, it becomes necessary for the network to react to this failure with one of several strategies:

- Local repair to overcome failed backbone nodes: Once a node detects failure of the next node in the backbone path to the sink, it initiates a local repair process, which bypasses the failed node. This can be done by initiating a bypass path discovery process. This discovery process must have the next node in the backbone that is following the failed node as the destination. The discovery message must use a fixed hop-based time-out timer to prevent the discovery process from flooding the whole network.
- Reinitiate the backbone discovery process: A backbone node failure message is sent from the detecting node to the source node to reinitiate a new backbone discovery message.

The first repair strategy would be cost-effective for large and long LSNs, while the second would not be costly in the case of relatively smaller and shorter LSNs. In addition, similar repair strategies can be used to repair paths from an SN to the nearest backbone node.

These strategies for mitigating and handling node failures and their effectiveness constitute a good basis for future research in this area.

**Algorithm 7:** LBDx

```
for i = 1, 2, . . . , x do
    use BD to discover the shortest path I-Ai from I to Ai
    use BD to discover the shortest path Ai-S from Ai to S
end for
for i = 1, 2, . . . , x do
    join I-Ai and Ai-S
end for
use NBND to discover paths from each NB node to the closest BN node in the discovered
backbone
```

**Algorithm 8:** LBD2

```
use BD to discover the shortest path I-A1 from I and A1
use BD to discover the shortest path A1-S from A1 and S
use BD to discover the shortest path I-A2 from I and A2
use BD to discover the shortest path A2-S from A2 and S
join I-A1 and A1-S
join I-A2 and A2-S
use NBND to discover paths from each NB node to the closest BN node in the discovered backbone
```

Figure 4 shows the different algorithms used in this paper along with their sequence of execution as the corresponding discovery messages are initiated and processed by the various node types.

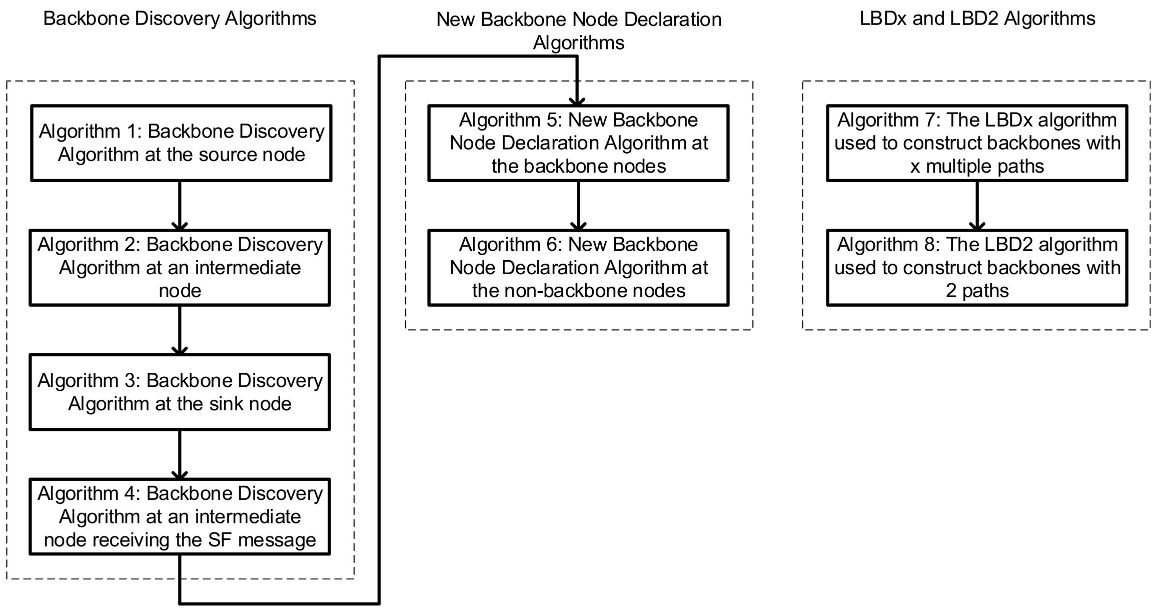

Backbone Discovery Algorithms
New Backbone Node Declaration Algorithms
LBDx and LBD2 Algorithms
Algorithm 1: Backbone Discovery Algorithm at the source node
Algorithm 2: Backbone Discovery Algorithm at an intermediate node
Algorithm 3: Backbone Discovery Algorithm at the sink node
Algorithm 4: Backbone Discovery Algorithm at an intermediate node receiving the SF message
Algorithm 5: New Backbone Node Declaration Algorithm at the backbone nodes
Algorithm 6: New Backbone Node Declaration Algorithm at the non-backbone nodes
Algorithm 7: The LBDx algorithm used to construct backbones with x multiple paths
Algorithm 8: The LBD2 algorithm used to construct backbones with 2 paths

**Figure 4.** Flowchart showing the sequence of execution and relationship of the various algorithms.

## 5. Performance Evaluation

### *5.1. Simulation Setup*

Since there was no real large-scale thick linear sensor network deployed at the time, we resorted to simulations to evaluate the performance of the proposed algorithms. In our simulations, we represented a thick linear wireless sensor network with a rectangular area, which represented the geographic span of a thick LSN. In our experiments, the length of the thick sensor network (i.e., the rectangular area) L was set to 10,000 m by default, and the width of the rectangular area (i.e., W) was set to 500 m by default. The number of sensor nodes (i.e., N) was set to 1000 by default. In our simulations, all the sensor nodes were uniformly distributed within the rectangular area. We assumed that the sensor nodes were homogeneous, and the communication range of each sensor node (i.e., the range) was set to 100 m by default. The broadcast ring size was set to $\frac{W}{2*\text{Range}} - 1$, which was 2 by default. These default settings followed several previous studies, and we believed these settings represented some typical scenarios that urgently needed efficient routing algorithms (e.g., LBDx).

Note that we may have varied one or several parameters mentioned above and left the other ones unchanged to investigate the impact of the corresponding parameters on the overall routing performance. Aside from that, in real applications, sensor nodes may be deployed in hills, waters, or environments with obstacles, which may result in sensor nodes not staying at the same height. To compensate for this, we measured the distance between two sensor nodes using the Euclidean distance between them in three-dimensional space instead of a two-dimensional plane. Usually, the obstacles will not affect the communication between sensor nodes; however, if this happened, we could vary the communication range of each sensor node to capture this interference.

### *5.2. Simulation Results*

*(1) LBD with Larger LSNs:* The performance of LBD in large networks is presented in Figure 5. Figure 5a shows that the time for discovery of the backbone decreased when the number of sensor nodes (N) increased. In addition, the decrease rate decreased as N increased. This was due to the fact that the larger number of sensor nodes increasingly did not help to provide more connectivity. On the other hand, Figure 5b,c shows that the number of LD + SF and NBD messages increased as N increased. Furthermore, it can be seen that the increase rate increased, and N increased as well. This is reasonable since these messages were transmitted by the nodes in a broadcast manner, which was proportional to the square of the number of sensor nodes.

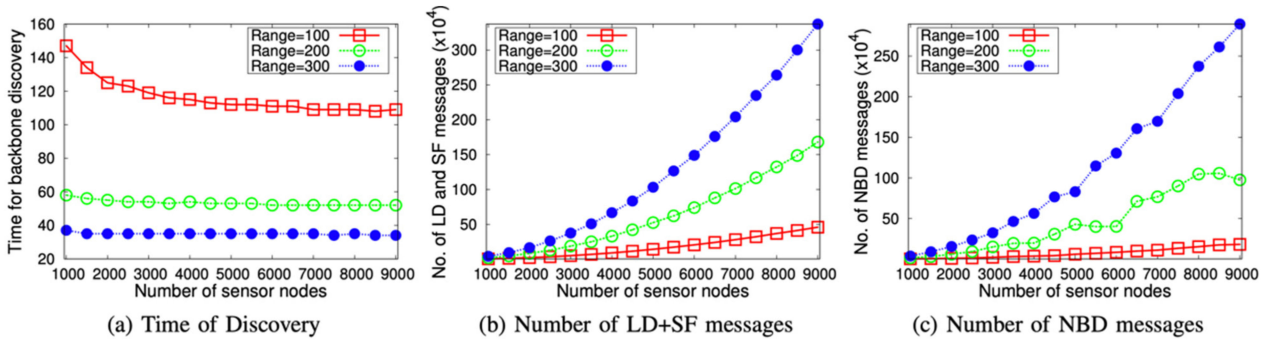
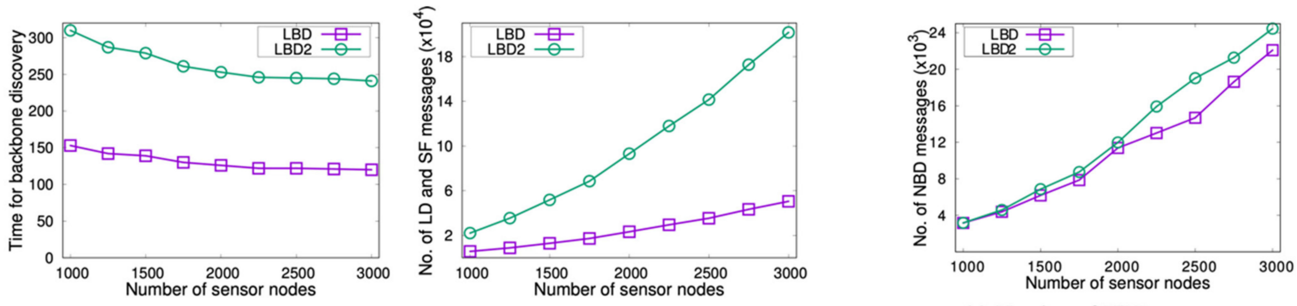


**Figure 5.** LBD with larger LSNs [39]. (**a**) Time of Discovery (**b**) Number of LD + SF messages (**c**) Number of NBD messages.

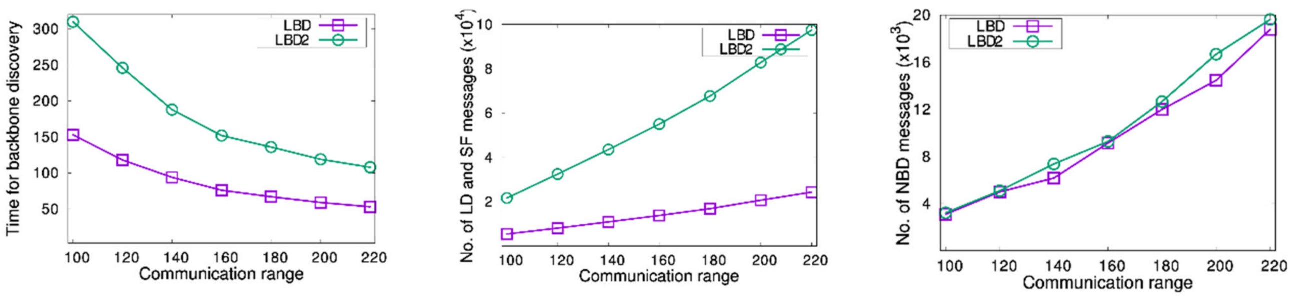
*(2) Comparison between LBD and LBD2:* In Figure 6, a comparison between LBD and LBD2 is shown, with the communication range set to 100 m. Figure 6a shows that the discovery time for LBD2 was approximately twice that of LBD as the number of sensor nodes increased. This is due to the fact that LBD2 constructs two consecutive paths. In other words, for each one of the parallel paths between the source and the sink, we constructed one part from the source to the middle anchor node and another part from the anchor node to the sink. On the other hand, LBD constructs only one path. In Figure 6b, we see that the number of LD + SF messages in LBD2 was four times that of LBD. This is reasonable, since LBD2 needs to find four path segments in order to discover the backbone, as opposed to only one path being required for LBD. In addition, relatively the same number of LD and SF messages were used to discover the shorter paths (or path segments) in LBD2 as well as the longer path in LBD. In Figure 6c, we see that the number of NBD messages was approximately the same for LBD2 and LBD. This is also reasonable to expect. Even though two long paths are discovered to construct the backbone in LBD2 as opposed to one long one in LBD, the newly discovered BN nodes can cover the NB nodes with paths that have a smaller number of hops.



(a) Time of Discovery (b) Number of LD+SF messages (c) Number of NBD messages

**Figure 6.** Performance results of the LBD and LBD2 algorithms as the number of sensor nodes increases, with the communication range fixed at 100 [39]. (**a**) Time of Discovery (**b**) Number of LD + SF messages (**c**) Number of NBD messages.

In Figure 7, we set the number of sensor nodes to 1000 and studied the performance of the algorithms when the communication range changed. Figure 7a shows that the backbone discovery time decreased as the communication range increased. This is reasonable to expect, since the BD and SF message sent from the source to the sink and back to the source propagated with a smaller number of hops. On the other hand, Figure 7b,c shows that the number of LD + SF messages varied considerably between the two algorithms, while the number of NBD messages stayed very close. In the two figures, we see that the number of messages in both algorithms increased along with the increased communication range. This is due to the fact that the increase in the communication range led to a higher number of sensor node pairs that could communicate with each other, resulting in a higher number of exchanged messages.



(**a**) Time of Discovery (**b**) Number of LD + SF messages (**c**) Number of NBD messages

**Figure 7.** Performance results of the LBD and LBD2 algorithms as the communication range increased with the number of sensor nodes fixed at 1000. (**a**) Time of Discovery (**b**) Number of LD + SF messages (**c**) Number of NBD messages.

*(3) Visual Example:* In Figure 8, we provide visual examples to show samples of the constructed backbone with the same LSN using the different algorithms, which include LBD and LBDx with x = 2, 3, and 4. In these examples, we used the following parameters. The number of sensor nodes was set to 300. The communication range was set to 200. The length and width of the LSN were set to 2500 and 500, respectively. The locations of the sensor nodes were generated randomly within the designated area. The locations of the anchor nodes were set as follows: in LBD2, A1 = (L/2, W/4) and A2 = (L/2, 3W/4); in LBD2, A1 = (L/2, W/4), A2 = (L/2, W/2), and A3 = (L/2, 3W/4); in L4BN, A1 = (L/4, 3W/4), A2 = (L/4, W/4), A3 = (3L/4, 3W/4), and A4 = (L/4, W/4). The figure shows that various LBDx algorithms generated more backbone paths. Consequently, the average number of hops from the NB nodes to the BN nodes decreased, which led to a subsequent decrease in the average number of hops from the NB nodes to the sink. This fact is illustrated in the next section in more detail.

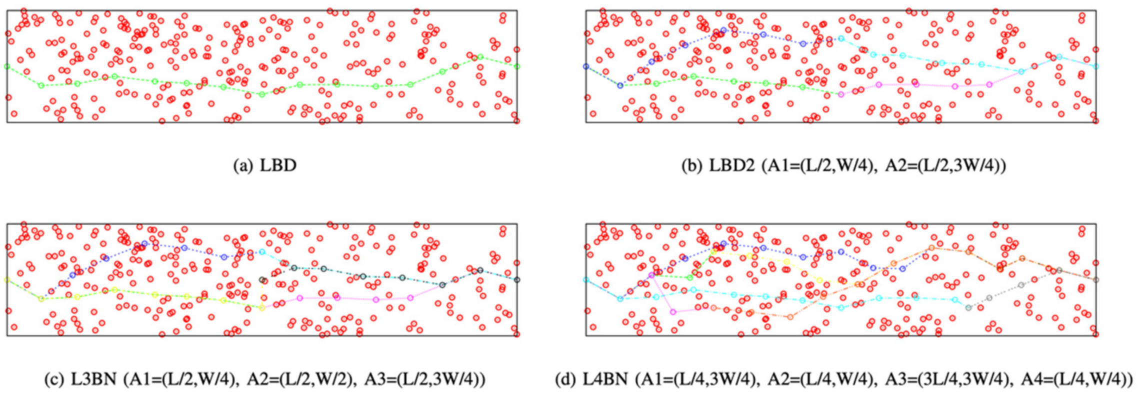

(a) LBD
(b) LBD2 (A1=(L/2,W/4), A2=(L/2,3W/4))
(c) L3BN (A1=(L/2,W/4), A2=(L/2,W/2), A3=(L/2,3W/4))
(d) L4BN (A1=(L/4,3W/4), A2=(L/4,W/4), A3=(3L/4,3W/4), A4=(L/4,W/4))

**Figure 8.** Visual examples of the discovered backbones for LBD, LBD2, LBD3, and LBD4, with L = 2500, W = 500, range = 200, and N = 300. (**a**) LBD (**b**) LBD2 (A1 = (L/2,W/4), A2 = (L/2,3W/4)) (**c**) LBN3 (A1 = (L/2,W/4), A2 = (L/2,W/2), A3 = (L/2,3W/4)) (**d**) LBN4 (A1 = (L/4,3W/4), A2 = (L/4,W/4), A3 = (3L/4,3W/4), A4 = (L/4,W/4)).

*(4) LBDx Advantage:* The main advantage of the LBDx algorithm over LBD is to reduce the number of hops to route data packets from NB nodes to the sink. The number of hops from a node X to the sink consisted of the sum of the number of hops from node X to the nearest node Y in the backbone and the number of hops from node Y to the sink. This is an important performance metric which indicates the amount of delay and total transmission energy consumed to send data packets from the different nodes in the LSN to the sink.

It is reasonable to expect that the number of communication hops for data message forwarding for LBD is higher than that of LBDx. The results of the simulation experiments presented in Figure 9 show the size of the gap between the two algorithms when the communication range was fixed at 100. It was observed that the average number of communication hops for LBD, which was near 15, was almost twice that of LBD2, which was near 6. The figure also shows the results for LBD3 and LBD4, which indicated an additional decrease in the average number of hops as the number of backbones x increased (i.e., three and four). However, the decrease margin (i.e., the decrease rate) did so as well. These results show that selecting the proper number of backbones for a particular LSN is an important configuration decision that should be done based on the related parameters, which include the network length, width, node density, and communication range. As we mentioned earlier, LBD tries to minimize the number of construction messages, and the simple strategy S*, in which each sensor node finds its shortest path to the sink, tried to minimize the average number of communication hops. Therefore, we proposed LBDx, which tries to have a trade-off between the number of construction messages and the average number of communication hops. The hyper parameter x in the proposed LBDx is used to control this trade off. Intuitively, the larger the parameter x is, the greater the number of construction messages is, and the lower the average number of communication hops is. Therefore, the exact value of the hyper parameter x is determined by the high-level application requirements rather than the topology of the underlying WSN.

Figure 10 further illustrates the advantage of LBD2 over LBD for larger LSNs. In the figure, the communication range was set to 100, and the number of sensor nodes was set to 1000. The figure shows the total number of message forwardings, versus the total number of normal data messages. The results show that when the total number of normal data messages exceeded 2000, the number message forwardings for LBD2 was less than that of LBD. The number of data messages exceeding that number is easily expected to be reached for larger LSNs with a number of sensor nodes over 1000.

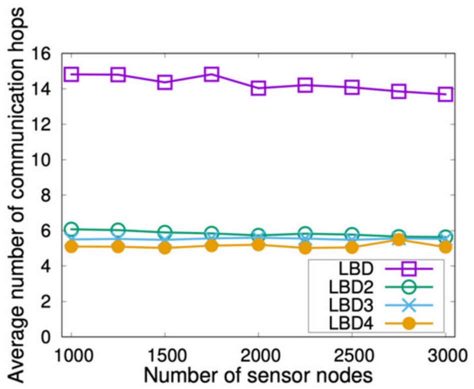


**Figure 9.** Comparison of the average number of hops for LBD, LBD2, LBD3, and LBD4 as the number of nodes increased.

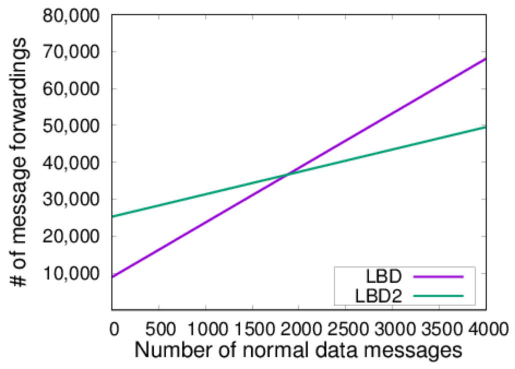


**Figure 10.** The number of message forwardings versus the number of normal data messages for LBD and LBD2.

### *5.3. Target Scenarios for LBD and LBDx*

Consequently, we can identify the target scenarios for choosing LBD and LBDx. On one hand, we see that the number of backbone discovery control message exchanges (i.e., LD and SF messages), as indicated in Figure 7, was generally larger in LBDx (the figure shows the results for x = 2). However, on the other hand, we see that the number of communication hops for subsequent data messages was significantly lower for LBDx compared with LBD. Therefore, we propose that LBDx be used in larger and ”thicker” LSNs (i.e., large LSN width W). In fact, as W increases, the number of backbone paths x should also be increased. This is because the overhead that is incurred due to the increase in the number of discovery messages with a larger x value would be well justified due to the significant reduction in the number of communication hops for the data messages (almost half for x = 2). Such data messages are expected to increase with larger and thicker LSNs. This reduction in the number of communication hops leads to smaller energy consumption by the individual sensor nodes that are involved in the forwarding process, which causes lower battery consumption and increases the average network lifetime. In addition, it is noted that the width of the LSN W is dictated by the application of the thick LSN. For example, border monitoring applications are expected to employ LSNs with larger widths where LBDx would be more appropriate. Furthermore, the value of x would be proportional to the LSN width W, while other applications such as roadside monitoring are expected to have LSNs with limited widths where LBD would be more appropriate.

## 6. Conclusions and Future Research

In this paper, we presented topology discovery algorithms for thick LSNs. The algorithms, LBD and LBDx, take advantage of the linearity of a network in order to increase the efficiency of the backbone discovery process. The discovered backbone can be used later for data transmission from all of the nodes in the LSN to the sink. The algorithms have desirable characteristics such as increased scalability, reliability, and fault tolerance. We also conducted simulation experiments in order to verify the operation of the algorithms and analyze their performance as various network parameters changed. Different algorithms and parameters can be used depending on the application involved. In the future, we intend to extend our work to further increase the fault tolerance by allowing the transmissions to jump over failed nodes. We will also consider various strategies for optimizing the energy consumption of the backbone nodes and provide efficient strategies for load balancing and handling node failures.

**Author Contributions:** Investigation, I.J., J.W. and N.M.; Software; Writing—original draft, I.J. and S.Z.; data curation/formal analysis S.Z.; Methodology I.J., J.W. and S.Z.; Writing—review & editing, I.J., J.W., N.M. and M.M.M.; funding acquisition I.J., N.M., and M.M.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded in part by UAEU—UPAR Grant No.: 31T059- UPAR (1) 2014 under Grant Code G00001655.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Jawhar, I.; Mohamed, N.; Agrawal, D.P. Linear wireless sensor networks: Classification and applications. *J. Netw. Comput. Appl.* **2011**, *34*, 1671–1682.
2. Jawhar, I.; Mohamed, N.; Al-Jaroodi, J.; Agrawal, D.P.; Zhang, S. Communication and networking of uav-based systems: Classification and associated architectures. *J. Netw. Comput. Appl.* **2017**, *84*, 93–108. [CrossRef]
3. Wang, Y. Topology Control for Wireless Sensor Networks. In *Wireless Sensor Networks and Applications*; Springer: Boston, MA, USA, 2008; pp. 113–147.
4. Jawhar, I.; Wu, J.; Mohamed, N.; Zhang, S. An efficient graph search algorithm for backbone discovery in wireless linear sensor networks. In Proceedings of the 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, Dallas, TX, USA, 22 October 2015; Institute of Electrical and Electronics Engineers (IEEE): San Diego, CA, USA, 2015; pp. 604–609.
5. Diggavi, S.N.; Grossglauser, M.; Tse, D.N.C. Even one-dimensional mobility increases ad hoc wireless capacity. In Proceedings of the IEEE International Symposium on Information Theory, Lausanne, Switzerland, 30 June–5 July 2002; p. 352.
6. Ghasemi, A.; Nader-Esfahani, S. Supporting aggregate queries over ad-hoc sensor networks. *IEEE Commun. Lett.* **2006**, *10*, 251–253. [CrossRef]
7. Miorandi, D.; Altman, E. Connectivity in one-dimensional ad hoc networks: A queueing theoretical approach. *Wirel. Netw.* **2006**, *12*, 573–587. [CrossRef]
8. Younis, M.; Senturk, I.F.; Akkaya, K.; Lee, S.; Senel, F. Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Comput. Netw.* **2014**, *58*, 254–283. [CrossRef]
9. Javadi, M.; Mostafaei, H.; Chowdhurry, M.U.; Abawajy, J.H. Learning automaton based topology control protocol for extending wireless sensor networks lifetime. *J. Netw. Comput. Appl.* **2018**, *122*, 128–136. [CrossRef]
10. Dhanapala, D.C.; Jayasumana, A.P. Topology preserving maps: Extracting layout maps of wireless sensor networks from virtual coordinates. *IEEE/ACM Trans. Netw. (TON)* **2014**, *22*, 784–797. [CrossRef]
11. Douik, A.; Aly, S.A.; Al-Naffouri, T.Y.; Alouini, M.S. Robust node estimation and topology discovery algorithm in large-scale wireless sensor networks. *arXiv* **2015**, arXiv:1508.04921.
12. Yu, T.; Wang, X.; Shami, A. Physical Topology Discovery Scheme for Wireless Sensor Networks Using Random Walk Process. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Institute of Electrical and Electronics Engineers, Washington DC, USA, 4–8 December 2016; pp. 1–5.
13. Kim, N.; Noel, E.; Tang, K.W. WSN communication topology construction with collision avoidance and energy saving. In Proceedings of the 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2014; Institute of Electrical and Electronics Engineers (IEEE): Las Vegas, NV, USA, 2014; pp. 398–404.

14. Ramanathan, R.; Rosales-Hain, R. Topology control of multihop wireless networks using transmit power adjustment. In Proceedings of the IEEE INFOCOM 2000; Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 26–30 March 2000; Institute of Electrical and Electronics Engineers (IEEE): Lausanne, Switzerland, 2002; Volume 2, pp. 404–413.
15. Shahabuddin, M.Z.; Hasbullah, H.; Aziz, I.A. Preliminary framework of topology control algorithm in WSN to achieve node's energy efficiency. In Proceedings of the Institute of Electrical and Electronics Engineers (IEEE), 3rd International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 15–17 August 2016; pp. 259–263.
16. Deb, B.; Bhatnagar, S.; Nath, B. A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management. 2001. Available online: https://www.researchgate.net/publication/215619114_A_Topology_Discovery_Algorithm_for_Sensor_Networks_with_Applications_to_Network_Management (accessed on 1 July 2021).
17. Bagci, H.; Korpeoglu, I.; Yazici, A. A Distributed Fault-Tolerant Topology Control Algorithm for Heterogeneous Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *26*, 914–923. [CrossRef]
18. Abu-Khzam, F.N.; Markarian, C.; auf der Heide, F.M.; Schubert, M. Approximation and Heuristic Algorithms for Computing Backbones in Asymmetric Ad-hoc Networks. *Theory Comput. Syst.* **2018**, *62*, 1673–1689. [CrossRef]
19. Kavitha, V.; Moorthi, M. A quality of service load balanced connected dominating set–stochastic diffusion search (CDS–SDS) network backbone for MANET. *Comput. Netw.* **2019**, *151*, 124–131. [CrossRef]
20. Saeedvand, S.; Aghdasi, H.S.; Khanli, L.M. Novel Distributed Dynamic Backbone-based Flooding in Unstructured Networks. *Peer-to-Peer Netw. Appl.* **2019**, *13*, 872–889. [CrossRef]
21. Kunz, T. Efficient Backbone Routing in Hierarchical MANETs. In Proceedings of the International Conference on Ad Hoc Networks, Paris, France, 17 November 2020; Springer: Cham, Switzerland, 2020; pp. 147–163.
22. Kumar, S.; Singh, A.K. A.K. A Backbone Formation Protocol Using Minimum Spanning Tree in Cognitive Radio Networks. In *Data Preprocessing, Active Learning, and Cost Perceptive Approaches for Resolving Data Imbalance*; IGI Global: Hershey, PA, USA, 2020; pp. 211–223.
23. Zhang, J.; Liu, S.-J.; Tsai, P.-W.; Zou, F.-M.; Ji, X.-R. Directional virtual backbone based data aggregation scheme for Wireless Visual Sensor Networks. *PLoS ONE* **2018**, *13*, e0196705. [CrossRef] [PubMed]
24. Luo, C.; Chen, W.; Yu, J.; Wang, Y.; Li, D. A novel centralized algorithm for constructing virtual backbones in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw. 2018* **2018**, *55*. [CrossRef]
25. Papakostas, D.; Eshghi, S.; Katsaros, D.; Tassiulas, L. Energy-aware backbone formation in military multilayer ad hoc networks. *Ad Hoc Netw.* **2018**, *81*, 17–44. [CrossRef]
26. Park, S.-Y.; Jeong, D.; Shin, C.S.; Lee, H. DroneNet+: Adaptive Route Recovery Using Path Stitching of UAVs in Ad-Hoc Networks. In Proceedings of the GLOBECOM IEEE Global Communications Conference, Institute of Electrical and Electronics Engineers (IEEE), Singapore, 4–8 December 2017; pp. 1–7.
27. Guo, J.; Liu, X.; Jiang, C.; Cao, J.; Ren, Y. Distributed fault-tolerant topology control in cooperative wireless ad hoc networks. In Proceedings of the IEEE Transactions on Parallel and Distributed Systems, Sydney, Australia, 14 October 2014; Volume 26, pp. 2699–2710.
28. Hajiaghayi, M.; Immorlica, N.; Mirrokni, V.S. Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks. In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, Los Angeles, CA, USA, 14 September 2003; pp. 300–312.
29. Santi, P. Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.* **2005**, *37*, 164–194. [CrossRef]
30. Tarique, M.; Tepe, K.E.; Adibi, S.; Erfani, S. Survey of multipath routing protocols for mobile ad hoc networks. *J. Netw. Comput. Appl.* **2009**, *32*, 1125–1143. [CrossRef]
31. Habib, A.; Saha, S.; Nur, F.N.; Razzaque, A. Starfish Routing for Wireless Sensor Networks with a mobile sink. In Proceedings of the 2016 IEEE Region 10 Conference (TENCON), Singapore, 22–25 November 2016; Institute of Electrical and Electronics Engineers (IEEE): Savannah, GA, USA, 2016; pp. 1093–1096.
32. Saha, S.; McLauchlan, L.; Saha, S. An energy balanced topology construction protocol for Wireless Sensor Networks. In Proceedings of the 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Las Vegas, NV, USA, 30 June–2 July 2014; Institute of Electrical and Electronics Engineers (IEEE): Sydney, Australia; pp. 1–6.
33. Sathya, S.; Sowmiya, P.; Vijayaraghavan, R.; Ragavi, A.; Rajkumar, G. Minimization of energy consumption in Wireless Sensor Networks using Virtual Backbone Scheduling with hierarchical clustering. In Proceedings of the International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, 13–14 February 2014; Institute of Electrical and Electronics Engineers (IEEE): Sydney, Australia, 2014; pp. 1–6.
34. Dhawan, A.; Tanco, M.; Yeiser, A. Randomized algorithms for approximating a Connected Dominating Set in Wireless Sensor Networks. In Proceedings of the International Conference on Computing and Network Communications (CoCoNet), Trivandrum, India, 16–19 December 2015; Institute of Electrical and Electronics Engineers (IEEE): San Diego, CA, USA, 2015; pp. 589–596.
35. Nimisha, T.S.; Ramalakshmi, R. Energy efficient connected dominating set construction using ant colony optimization technique in Wireless Sensor Network. Proceedings of International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 19–20 March 2015; IEEE: San Diego, CA, USA, 2015; pp. 1–5.

36. Jawhar, I.; Zhang, S.; Wu, J.; Mohamed, N.; Masud, M.M. Efficient topology discovery and routing in thick wireless Linear Sensor Networks. In Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 1–4 May 2017; Institute of Electrical and Electronics Engineers (IEEE): Singapore, 2017; pp. 91–96.
37. Shu, L.; Zhang, Y.; Yang, L.T.; Wang, Y.; Hauswirth, M.; Xiong, N. TPGF: Geographic routing in wireless multimedia sensor networks. *Telecommun. Syst.* **2009**, *44*, 79–95. [CrossRef]
38. Alafeef, I.; Awad, F.; Al-Madi, N. Energy-aware geographic routing protocol with sleep scheduling for wireless multimedia sensor networks. In Proceedings of the 14th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT), Irbid/Amman, Jordan, 9–11 October 2017; Institute of Electrical and Electronics Engineers (IEEE): Singapore, 2017; pp. 93–97.
39. Gopi, P. Multipath Routing in Wireless Sensor Networks: A Survey and Analysis. *IOSR J. Comput. Eng.* **2014**, *16*, 27–34. [CrossRef]