Efficient Topology Discovery and Routing in Thick Wireless Linear Sensor Networks

Imad Jawhar[†], Sheng Zhang[‡], Jie Wu[§], Nader Mohamed[◇], and Mohammad M. Masud[♯]
[†]Midcomp Research Center, Saida, Lebanon
[‡]State Key Laboratory for Novel Software Technology, Nanjing University, China
[§]Department of Computer and Information Sciences, Temple University, USA
[◇]Middleware Technologies Labs, Isa Town, Bahrain
[♯]College of Information Technology, UAE University, Alain, UAE
Emails: imad@midcomp.net, sheng@nju.edu.cn, jiewu@temple.edu, nader@middleware-tech.net, m.masud@uaeu.ac.ae

***Abstract*—Wireless devices such as sensors have increasingly more processing, storage, and networking capabilities, making wireless sensor networks (WSNs) get lots of attentions in recent years. In addition, the cost of sensors is constantly decreasing making it possible to use large quantities of these sensors in a wide variety of important applications in environmental, military, commercial, health care, and other fields. In order to monitor certain types of infrastructures, many of these applications involve lining up the sensors in a linear form, making a special class of these networks which are defined as Linear Sensor Networks (LSNs). In this paper, we take advantage of the linearity of the network to design two graph-search-based topology discovery algorithms for LSNs, namely, LNBN and L2BN. LNBN focuses on minimizing the number of messages used to construct the backbone, while L2BN targets to minimizing the average number of communication hops. The proposed algorithms have several good properties. First, they allow for significant improvement in the scalability of the communication process. Second, the linearity of the structure and the discovered backbone can enhance the routing reliability by jumping over failed nodes by increasing the range. Lastly, they do not require sensor nodes to have location detection capabilities such as GPS, which would otherwise lead to higher costs of sensor nodes.**

***Index Terms*—Wireless linear sensor networks, backbone discovery, routing.**

I. Introduction

Wireless Sensor Networks (WSNs) have received a lot of attention due to constant advancements in the field of electronics and wireless communication leading to the design of low cost, small, and capable sensing devices with increasingly higher processing, storage, sensing and communication capabilities. In addition, WSNs have a great potential for use in a large amount of existing and future applications in numerous areas such as environmental, civil, health care, military, monitoring, and infrastructure surveillance. In the latter category, a considerable number of the infrastructures that are monitored have a linear structure which extends over relatively long distances. This causes the wireless sensors to be aligned in a linear topology. New frameworks and protocols are needed to take better advantage of the linearity of the network structure in order to increase routing efficiency, enhance reliability and security, and improve location management [1].

This work was supported in part by UAEU - UPAR Grant No.: 31T059-UPAR (1) 2014 under Grant Code G00001655.

We observed that, LSNs are thick in many scenarios. For example, an LSN can have the responsibility of monitoring international borders between countries [2] and detect illicit activities. Such activities can involve border crossings by smugglers, military crossings, etc. The inexpensive sensors can be deployed by throwing them from an airplane or an unmanned aerial vehicle (UAV). The dropped sensors end up in a semi-random geographic form and could follow a linear structure. The sink nodes can also be deployed at various locations and are separated by some specified average distance. The sink nodes could be thrown from a low-flying airplane, placing them at locations which are separated by approximately the same average distance, or they can be installed [3] in a precise fashion by the network personnel if the terrain is easily accessible. Applications for linear sensor networks include but are not limited to the following: above-ground oil, gas, and water pipeline monitoring; underwater oil, gas, and water pipeline monitoring; railroad/subway monitoring; terrestrial border monitoring; sea-coast monitoring; and river monitoring.

In this paper, we introduce two topology discovery algorithms, LNBN and L2BN, for thick LSNs, where the sensor nodes are deployed between two parallel lines that can stretch for a long distance (e.g. tens or hundreds of kilometers). LNBN and L2BN have different objectives: LNBN concentrates on minimizing the number of backbone construction messages [4], while L2BN focuses on minimizing the average number of communication hops. As a result of the proposed topology discovery algorithms, a small percentage of the deployed sensor nodes are selected to form a backbone network along the linear topology, which can be used to efficiently route sensing data along the linear network to the sink or sinks located at the end of the network or network segment. The performance of the proposed algorithms is evaluated by extensive simulations.

The rest of the paper is organized as follows. Section II discuss related work. Section III presents LNBN. Section IV presents L2BN. Section V provides simulation results, and section VI concludes the paper.

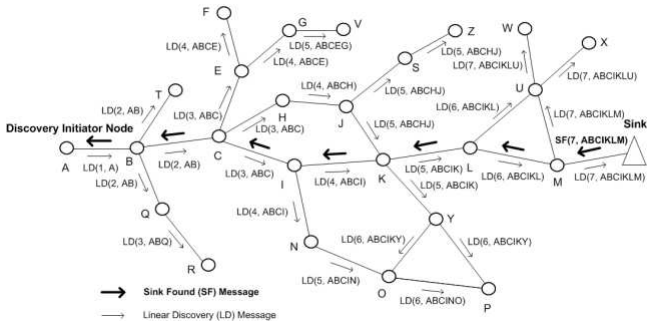


Fig. 1: Illustration of the LD message propagation from the initiator node to the sink in the Linear Backbone Discovery (LBD) algorithm.

II. Related Work

One-dimensional (1-D) ad hoc networks have been studied by various researchers. Diggavi et. al. studied the characteristics of wireless capacity with 1-D mobility [5]. Ghasemi et al. provided an approximation formula for the connectivity probability of 1-D ad hoc wireless networks [6]. Miorandi et al. analyzed the connectivity issue in 1-D ad hoc networks using a queuing theory approach [7].

On the other hand, many researchers have investigated topology control techniques in wireless ad hoc networks. In [8], Santi et al. present a comprehensive survey. In [9], Ramanathan et al. study the optimization problem of creating a desired topology by adjusting the transmit power of the nodes. In another paper [10], the authors study power assignments to maintain fault tolerance in wireless devices and present algorithms which can be used to minimize power while maintaining k-edge connectivity with guaranteed approximation factors. In [11], a topology discovery algorithm for WSNs is presented. The algorithm determines a set of nodes which can act as cluster heads in the network. In [3], Wang presents an overview of the different types of topology algorithms for multidimensional WSNs that have been proposed in research.

These algorithms are primarily designed for multi-dimensional WSNs. They do not take advantage of the predictable topology of a thick LSN in order to optimize their performance. Different from them, the algorithms presented in this paper are designed to take advantage of the linearity of the network in order to reduce topology discovery control overhead, and increase communication efficiency.

III. Topology Discovery Algorithm: LNBN

LNBN consists of two phases: linear backbone discovery (LBN) and new backbone node declaration. LNBN tries to construct a backbone for a thick WSN and minimizes the number of messages used.

A. Linear Backbone Discovery (LBD) Algorithm

The LBD algorithms is shown in Algorithms 1, 2, 3, and 4. It is also illustrated in Figure 1. It works in the following manner. As indicated in Algorithm 1, the designated first node on the primary edge of the LSN starts the discovery process by initializing its discovery variables and broadcasting the Linear Discovery (LD) message $LD(messageID, myID, messageLc, PATH)$ to all of its neighbors. The message contains the following parameters: $messageID$, the ID or the discovery message to prevent looping; $myID$, the ID of the sending node; $messageLc$, the linear discovery counter, which holds the count of nodes in the discovered path from the initial primary edge node that initiated the discovery process; and $PATH$, an ordered list of nodes that are contained in the discovered path.

Algorithm 1 Backbone Discovery - Broadcasting the LD message from the 1st node at the primary edge

$myTempParent$ = $myConfParent$ = ϕ
if (this is the first node at the primary edge) **then**
  $myLc$ = 0, $PATH$ = $myID$, $messageLc$ = 1
  $SendLD(messageID, myID, messageLc, PATH)$ to all neighbors
else
  $myLc$ = ∞
end if

Algorithm 2 Backbone Discovery - Algorithm at an intermediate node y when receiving an LD message from x

When node y receives the $LD(messageID, x, messageLc)$ from node x it does the following:
if ($messageLc < myLc$) **then**
  $myTempPrarent$ = x, $myLc$ = $messageLc$
  $messageLc$ = $messsageLc$+1, $PATH$ = $PATH \mid myID$
  Broadcast $LD(messageID, myID, messageLc , PATH)$
else
  Drop LD message
end if

Algorithm 2 describes the actions executed by an intermediate node y when it receives an LD message from node x. First node y checks to see if the linear counter in the message $messageLc$ is better (i.e. smaller) than its own linear counter $myLc$. If that is the case, then it changes its temporary parent to x, and updates its $myLc$ counter with that, which is in the message. It then increments the linear counter in the message by one, adds its own ID to the $PATH$ and broadcasts the updated LD message to all of its neighbors. However, if $messageLc$ is not smaller than $myLc$, then it drops the message. Because it already has a parent node with a better linear count with a smaller number of hops from the source, which contributes with a lower number of hops in the backbone.

Algorithm 3 describes the actions taken by the sink when it receives an LD from a node x. Namely, it saves the ID of x as its backward neighbor as well as the length of the discovered backbone in number of hops that is contained in the $messageLc$. It then unicasts a sink found message, $SF(messageID, source = myID, destination = x, BBlc, PATH)$ back to the discovery initiating node through the nodes in the discovered backbone.

Algorithm 4 describes the steps taken by an intermediate node y when it receives an $SF(messageID, source = myID, destination = x, BBlc, PATH)$ message from a

Algorithm 3 Backbone Discovery - Algorithm at the sink when receiving an LD message from node x

When the sink receives the $LD(messageID,\ x,\ messageLc,\ PATH)$ message from node x it does the following:
$myBacwadNeigh$=x, $BBLc$ = $messageLc$
Send $SF(messageID,\ source = myID,\ destination = x,\ BBlc,\ PATH)$

Algorithm 4 Backbone Discovery - Algorithm at an intermediate node y when receiving a SF message from node x

$iAmPartOfBackbone$ = $TRUE$
Save the full or local part of the discovered backbone in $PATH$ in the routing table according to the adopted caching strategy.
$myBackwardDirNeigh$ = $myTempParent$
$myForwardDirNeigh$ = x
$myDistFromSource$ = $messageLc$
$myDistFromSink$ = $messageLc$ - $myLC$
Send $SF(messageID,\ source = myID,\ destination = x,\ messageLc,\ PATH\ )$

node x. First, y sets its $iAmPartOfBackbone$ variable to $TRUE$. Then node y fully or partially caches the discovered backbone depending on the strategy that is used. A full caching of the backbone allows the node to have the full list of the nodes in the backbone and consequently node y has more flexibility in routing packets. However, this comes at the cost of increased memory usage. On the other hand, node y can partially cache the local part of the backbone such as k nodes in each direction, which allows it some flexibility in routing packets and reaction to neighboring node failures while reducing its memory usage. Node y then sets its forward and backward direction neighbors, as well as the distance from the source, and distance from the sink in number of hops. Afterwards, node y forwards the SF message to its backward neighbor. This propagation of the SF message continues along the nodes in the discovered backbone till it reaches the source node, thereby completing the backbone discovery process.

At the end of the backbone discovery process, we will have two types of nodes: *Backbone Nodes (BNs)*, which are a part of the backbone, and *Non-backbone Nodes (NBs)*, which did not end up being a part of the backbone. They are used to perform normal sensing operations.

B. The New BN Declaration (NBD) Broadcast algorithm

At the end of the backbone discovery process, the newly discovered BN nodes will broadcast a a New BN Declaration (NBD) message to inform all of the nodes within ρ hops from itself that it is a part of the backbone. Algorithm 5 is used by the BN node to initiate the broadcast process of the NBD message. In the NBD message, the BN node includes the following parameters: $messageID$, which contains the message ID to prevent looping; $sourceBNID$, which is the ID of the sending BN node; $myID$, which is the ID of the node forwarding the BND message and is initially equal to the $sourceBNID$; $BNDringSize$, which is the size of the broadcast ring in number of hops and is set to

Algorithm 5 NBD Initiation - Algorithm initiated by a newly declared BN node

$sourceBNID$ = $myID$, $NBDringSize$ = ρ
$numOfHops$ = 0, $PATH\_TO\_BN$ = $myID$
Broadcast NBD $(messageID,\ sourceBNID,\ myID,\ NBDringSize,\ numOfHops,\ PATH\_TO\_BN)$

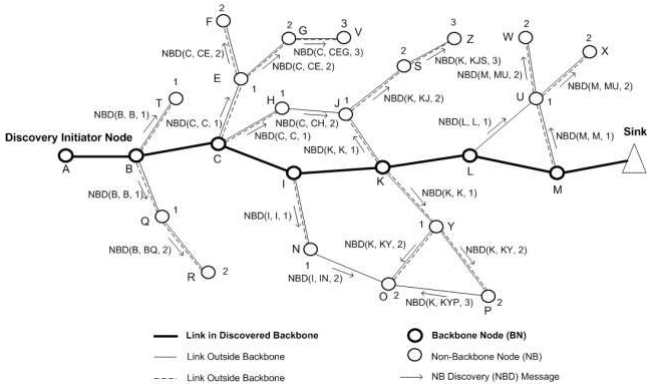


Fig. 2: Illustration of the NBD message propagation in the nearest BN node discovery algorithm.

ρ; $numOfHops$, which contains the number of hops that this message has traversed so far (it starts at 0 and is incremented as the BND message propagates through the nodes); and $PATH\_to\_BN$, which is the path to the BN node that is discovered so far. As the BND message is propagated, each intermediate node concatenates its own ID to the end of the $PATH\_to\_BN$ it received in from the previous node.

Algorithm 6 describes the steps taken by an intermediate node y when it receive the NBD message from another node x. Namely, when the NBD message reaches a node, it does the following. It caches the path, $PATH\_TO\_BN$, to the newly discovered BN node. Node y now can use this path to send messages to the BN node in order to transmit them to the sink through the backbone. It then increments the number of hops. If the new number of hops in the message is still less than or equal to the $ringSize$, then it adds its own ID to $PATH\_TO\_BN$, and broadcasts the message to its neighbors. Otherwise, it drops the message. Figure 2 provides an illustration of the NBD message propagation. As the SF message propagates back from the sink, each of the newly discovered BN nodes broadcasts an NBD message, which is initiated and propagated according to Algorithms 5 and 6 respectively. The figure shows the BN nodes, which are nodes A, B, C, I, K, L, and M. These nodes constitute the discovered backbone. It also shows the NB nodes, which were not designated as part of the backbone. Each of the NB nodes is shown with the corresponding distance (in number of hops) from the nearest BN node. The dashed lines show the path of each of the NB nodes to the nearest BN node according the the described algorithms. These paths are discovered after the broadcast and propagation of the NBD messages from the BN nodes.

Algorithm 6 NBD Propagation - Algorithm at an intermediate node y when receiving a NBD message from node x.

When node y receives an NBD ($messageID$, $sourceBNID$, $myID$, $BNDringSize$, $numOfHops$, $PATH\_TO\_BN$) from a node x.
save $PATH\_TO\_BN$ in the routing table as a path to the $sourBNID$ node, which is now a part of the backbone
$numOfHops$ = $numOfHops$ + 1
if ($numOfHops \leq ringSize$) **then**
  $PATH\_TO\_BN$ = $PATH\_TO\_BN \mid myID$
  Broadcast NBD ($messageID$, $sourceBNID$, $myID$, $BNDringSize$, $numOfHops$, $PATH\_TO\_BN$) message to all neighbors
else
  Drop NBD message
end if

IV. Topology Discovery Algorithm: L2BN

In the last section, we have proposed a distributed backbone discovery algorithm based on the shortest path between the discovery initiator node and the sink. This path is then used as the backbone path and the other non-backbone nodes send messages to the sink via the backbone path. Before introducing L2BN, let us examine some design alternatives.

When designing such kind of backbone discovery algorithms, we are often interested in two metrics: the number of messages generated for constructing the backbone (the number of construction messages for short), and the average number of hops for each sensor node to send messages to the sink (the average number of communication hops for short).

Here is another strategy: let every sensor node other than the sink send LD messages to the sink, i.e., each sensor node finds its shortest path to the sink. Of course, this strategy achieves the smallest average number of communication hops, however, it may incur unaffordable flooding message overhead.

In fact, LNBN and the above strategy represent two extremes in the design space: LNBN targets to minimize the number of construction messages while does not explicitly minimize the average number of communication hops, and SBN does in the reverse way.

Inspired by this observation, we thereby propose an algorithm that carefully balances this two design extremes. The basic idea is to construct two paths from the initiator to the sink (in contrast to selecting the shortest path in the LNBN algorithm). Although we may use more construction messages than LNBN, the average number of communication hops must decrease. We call this strategy L2BN. L2BN requires four anchor nodes: I, the discovery initiator, S, the sink, and two others. Suppose we can denote the thick WSN by a rectangle with length L and thickness T, and the top-left corner by (0,0). Then, the other two anchor nodes U and V can be represented by $(L/2, T/4)$ and $(L/2, 3T/4)$, respectively. We note that, most WSNs have anchor nodes for the localization purpose; besides, deploying anchor nodes is not hard and their positions can be flexibly adjusted for future unforeseen use.

The details of L2BN are shown in Alg. 7. First, we employ

Algorithm 7 L2BN

Require: I, S, A1, A2
1: use LBN to find the shortest path I-U between I and U
2: use LBN to find the shortest path U-S between A1 and S
3: use LBN to find the shortest path I-V between I and V
4: use LBN to find the shortest path V-S between V and S
5: catenate I-U and U-S
6: catenate I-V and V-S
7: use NBN for the other nodes to find their shortest paths to the constructed two backbone paths

LBN that relies on LD and SF messages to find the shortest paths from I to U, from U to S, from I to V, and from V to S. Then, we catenate I-U and U-S to generate one path, and catenate I-V and V-S to generate another one. Note that, these two paths are not necessarily node-disjoint. Lastly, we use NBN to declare these new backbone nodes and ensure the other non-backbone nodes to find their shortest paths to the two backbone paths.

It is worth noting that, L2BN is highly flexible: depending on the density of sensor nodes, the ratio of thick WSN height to width, and the relative importance of communication delay, we can adjust the number of anchor nodes to generate the backbone paths.

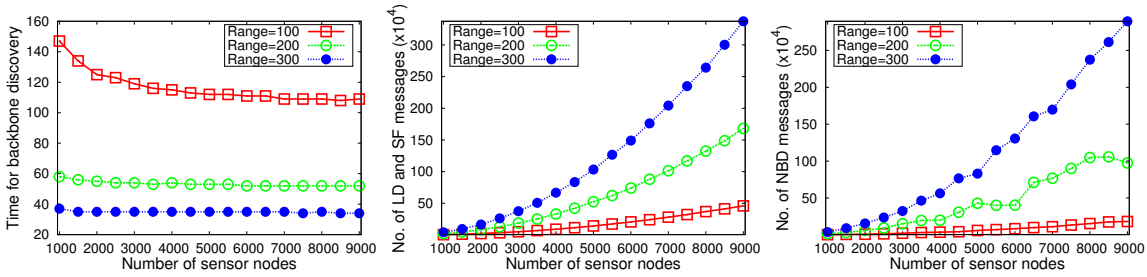
V. Performance Evaluation

A. Simulation Setup

The thick linear sensor network is generated according to the model stated in Sections I and II. A thick LSN is modeled as a rectangle in our simulations. Key parameters in the simulations include the thickness (i.e. the width) of the thick LSN, the length of the thick LSN, the number of sensor nodes, the communication range of a sensor node, and the size of the broadcast ring ρ. In our simulation, the default values of these input parameters are set as follows: the width W is 500 meters, the length L is 10000 meters, the number N of sensor nodes is 1000, the communication range $Range$ of each sensor node is 100 meters, the default ring size ρ is $\frac{W}{2Range} - 1$, which is equal to 2.

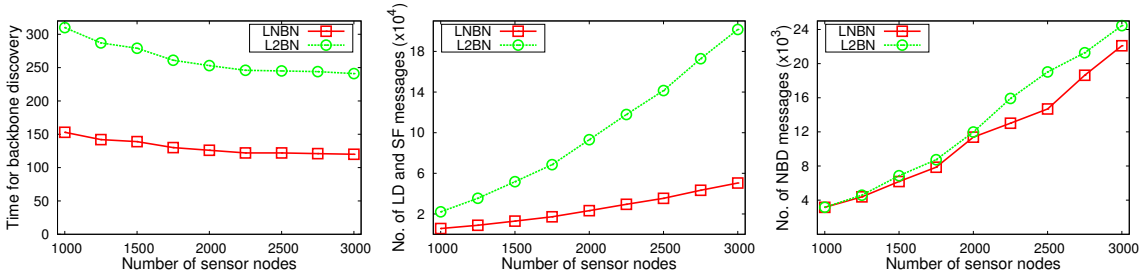
In all simulations, the position of each sensor node is uniformly generated within the 2-dimensional rectangle that represents the thick LSN. Two sensor nodes can communicate if and only if the distance between them is not larger than the communication range. The node that initiates the backbone discovery is the leftmost node within the 2-D rectangle; similarly, the sink is the rightmost node within the thick LSN.

The performance metrics used in our evaluations are the time for backbone discovery, the number of LD and SF messages used in the backbone discovery process, and the number of new backbone node declaration (NBD) messages. Our simulation seeks to investigate the impacts of these parameters. Thus, we ran experiments with one varying parameter while keeping the others to their default values. Each experiment run lasts for sufficiently long time, so as to better reflect the performance of the proposed algorithm.



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

Fig. 3: LNBN on large instances



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

Fig. 4: Comparison results of LNBN and L2BN under varying number of sensor nodes while fixing the communication range at 100

B. Simulation Results

1) LNBN on Large Networks: Fig. 3 presents how LNBN performs on large networks. We see in Fig. 3(a) that, when the number of sensor nodes (i.e., N) increases, the time of backbone discovery decreases, and its decreasing speed also decreases. This is due to the relatively large number of sensor nodes: more sensor nodes will not help improve the connectivity of the network. In Figs. 3(b) and (c), we see that, the number of LD+SF and NBD messages increases as the number of sensor nodes increases; increasingly, the increasing speed also increases. This is due to the fact that, these messages are spread in a broadcast way, and the number of message thereby increases with a speed that is proportional to the square of the number of sensor nodes.

2) Comparison Results of LNBN and L2BN: We are interested in comparing LNBN and L2BN. Fig. 4 shows the comparison results between LNBN and L2BN, in which the communication range is set to 100. In Fig. 4(a), we see that, the time of backbone discovery in L2BN is nearly twice as that in LNBN. This is because, we need to construct two paths in a distributed way—using messages—in L2BN. In Fig. 4(b), we find that, the number of LD+SF messages in L2BN is roughly four times as that in LNBN. This is reasonable, since L2BN uses LBN four times to find the corresponding backbone paths, and in each time, although LBN is used to find a "shorter" path, it performs the same operations as for a "longer" path. In this sense, the number of messages it may produce is four times as many as those in LNBN.

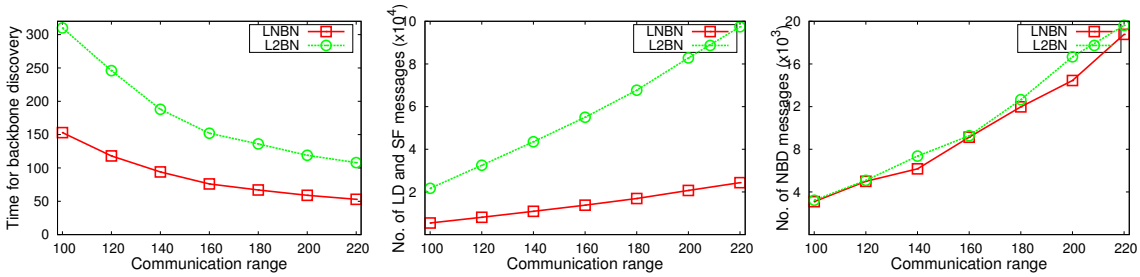
In Fig. 4(c), the number of NBD messages in L2BN is almost the same as that in LNBN. This is reasonable, since there are two backbone paths in L2BN that may produce more NBD messages, but these two paths can also cover the other sensor nodes in a shorter distance.

We also fix the number of sensor nodes at 1000, and see how LNBN and L2BN perform under varying communication range. Fig. 5(a) shows the comparison results. It is reasonable to see that, the time of backbone discovery decreases as the communication range increases. We also find that, the numbers of NBD messages in two algorithm are almost the same, while the number of LD+SF messages vary a lot. When the communication range increases, the number of pairs of sensor nodes that can communicate increases, therefore, the number of messages increases as well.

3) Visualization Example: We also provide in Fig. 6 two visualization examples of running LNBN and L2BN on the same underlying WSN. In this set of figures, we set the number of sensor nodes to be 300, the communication range to be 200, the width and the length of the thick WSN to be 500 and 2500, respectively. The positions of sensor nodes are randomly generated within the area. We see that, L2BN finds two backbone paths, making it cover the other non-backbone nodes in a "shorter" distance, and it is reasonable to conclude that, the average number of communication hops in L2BN is much smaller than that in LNBN, as we will shortly see in the next subsection.

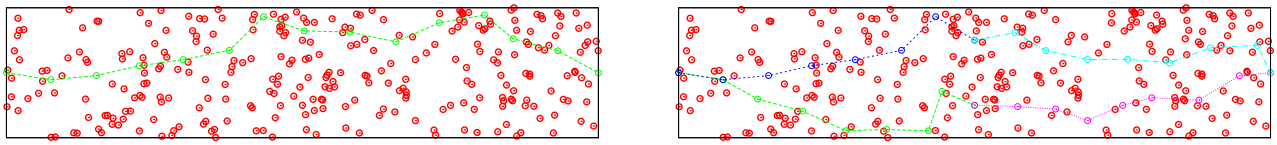
4) L2BN Benefit: One main purpose of backbone construction is to facilitate delivering normal data packets. We define the number of communication hops of a sensor node X as the sum of the number of hops of the shortest path from X to any node, say Y, in the backbone path and the number of hops from Y to the sink. This metric is very important, as it determines the delay of delivering normal data packets and the number of message forwardings.

It is important to note that, the average number of communication hops in L2BN is much smaller than that in LNBN. But how much? Fig. 7 shows the gap, where the communication



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

Fig. 5: Comparison results of LNBN and L2BN under varying communication range while fixing the number of sensor nodes at 1000



(a) LNBN (b) L2BN

Fig. 6: Illustration of the backbone path where $W = 500$, $L = 2500$, $N = 300$, and $Range = 200$.

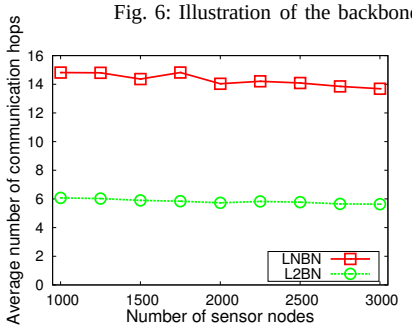


Fig. 7: Average number of communication hops

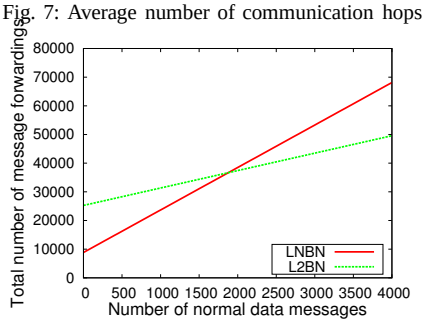


Fig. 8: Total number of message forwardings.

range is set to 100. We see that, the average number of communication hops in LNBN is roughly 15, which is about twice as many as that in L2BN, which is around 6.

The benefit of shorter communication path of L2BN will be more evident when we look at Fig. 8. We fix the number of sensor nodes at 1000, and the communication range at 100. The x-coordinate represents the number of normal data packets send; the y-coordinate represents the total number of message forwardings. We see that, when the number of normal data messages exceeds 2,000, the total number of message forwardings in L2BN becomes less than that in LNBN. We note that, considering the number of sensor nodes in the WSN of interest of interest to be 1,000, the number of normal data messages can easily exceed 2,000.

VI. Conclusions and Future Research

In this paper, we present two graph-search-based algorithms for backbone discovery in thick LSNs. The resulting backbone can be used for efficient data routing. The proposed algorithms have several good properties, namely, good scalability, increased reliability and fault tolerant. Our future work will focus on utilizing the linearity structure to allow the routing protocol to overcome node failures by jumping over failed nodes.

References

[1] I. Jawhar, N. Mohamed, and D. P. Agrawal, "Linear wireless sensor networks: Classification and applications," *Elsevier Journal of Network and Computer Applications*, vol. 34, pp. 1671–1682, 2011.

[2] A. D'Costa, V. Ramachandran, and A. M. Sayeed, "Distributed classification of gaussian space-time sources in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 1026–1036, August 2004.

[3] Y. Wang, "Topology control for wireless sensor networks," *Wireless Sensor Networks and Applications, Springer*, pp. 113–140, 2008.

[4] I. Jawhar, J. Wu, N. Mohamed, and S. Zhang, "An efficient graph search algorithm for backbone discovery in wireless linear sensor networks," in *Proc. of MiSeNet 2015*, pp. 604–609, Oct 2015.

[5] S. Diggavi, M. Grossglauser, and D. Tse, "Even one-demensional mobility increases adhoc wireless capacity," *IEEE Transactions on Information Theory*, vol. 51, November 2005.

[6] A. Ghasemi and S. Nader-Esfahani, "Supporting aggregate queries over ad-hoc sensor networks," *IEEE Communications Letters*, vol. 10, pp. 251–253, April 2006.

[7] D. Miorandi and E. Altman, "Connectivity in one-dimensional ad hoc networks: a queuing theoretical approach," *Wireless Networks*, vol. 12, pp. 573–587, September 2006.

[8] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Computing Surveys*, vol. 37, pp. 164–194, March 2005.

[9] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," *In Proc. of IEEE Infocom 2000*, pp. 404–413, March 2000.

[10] M. Hajiaghayi, N. Immorlica, and V. Mirrokni, "Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks," *in Proc. of ACM MobiCom 2003*, September 2003.

[11] B. N. B. Deb, S. Bhatnagar, "A topology discovery algorithm for sensor networks with applications to network management," *IEEE CAS workshop*, September 2002.