Data Transmission using Continuous Neighbor Discovery in Asynchronous Sensor Networks

M. Parameshwar  
Assistant professor, Department of CSE  
CMR College of Engineering & Technology, Hyderabad  
Email: mparameswar@gmail.com

G. Ravi Kumar  
Assistant professor, Department of CSE  
CMR College of Engineering & Technology, Hyderabad

I. Haripriya  
Assistant professor, Department of CSE  
CMR College of Engineering & Technology, Hyderabad

Abstract – In sensor networks the nodes are static. Yet, node connectivity is subject to changes owing to disruptions in wireless communication, transmission power changes, or loss of synchronization between neighboring nodes. Hence, even after a sensor is conscious of its immediate neighbors, it should endlessly maintain its view, a method we call continuous neighbor discovery. In this work we tend to distinguish between neighbor discovery during sensor network initialization and continuous neighbor discovery. We tend to target on the latter and consider it as a joint task of all the nodes in each connected segment. Each sensor employs a simple protocol in a very coordinate effort to scale back power consumption while not increasing the time needed to find hidden sensors. An efficient continuous neighbor discovery algorithm is employed to find the neighbor nodes.

This algorithm is predicted on detecting all hidden links within a segment. Namely, if a hidden node is discovered by one amongst its segment neighbors, it is discovered by all its other segment neighbors after a very short time. Hence, the discovery of a new neighbor is viewed as a joint effort of the whole segment. Existing system has some of the disadvantages like serious traffic, long term process, and greater expense of energy. Therefore, we tried to overcome all these drawbacks and that we compared it with the initial discovery.

Keywords – Neighbor Discovery, Sensor Networks.

I. INTRODUCTION

The sensor network contains very large number of sensor nodes. These sensor nodes may be connected to each other inside a network by any mesh structure. These sensor nodes can sense various events very sensitively. Some of the sensor nodes act as routers and gateways to pass the message or the file from one particular sensor node to another sensor node. In order to pass the data there will be high consumption of bandwidth, energy and even power. Therefore, we design this project in such a way that we can minimize these three critical issues. These issues can be overcome by alternatively putting the sensor nodes in active state and passive state. In this paper the sensor nodes are randomly distributed over a particular area and each sensor node has certain transmission area to cover. First we are detecting the immediate neighbors. That is called as initial discovery. The sensor nodes should have direct wireless communication between them. Then the sensor nodes should establish the particular shortest roots through which they can communicate with the other sensor nodes via any router or gateway in between.

The sensor nodes will be awake for a very short period of time, that is in a sensor network the nodes are exist for a while and after that they can change their positions or move away from the network. This paper presents a special neighbor discovery scheme that can be used to reduce the traffic that is being caused by the sensor nodes. Another important issue in the sensor network is that the sensor nodes despite of being static can change due to the following situation.

- Loss of local synchronization due to accumulated clock drifts.
- Disruption of wireless connectivity between adjacent nodes by a temporary event, such as a passing vehicle or animal or anything, a dust storm, rain or fog. When these events are over, the hidden nodes must be rediscovered.
- The ongoing addition of new nodes, in some networks to compensate for nodes which have ceased to function because their energy has been exhausted.
- Sometimes the nodes may move away from the network and loses the connection with the other nodes present in a network.
- The increase in transmission power of some nodes, in response to certain events, such as detection of Emergent situations.

For these reasons, detecting new links and nodes in sensor networks must be considered as an ongoing process. In the following discussion we distinguish between the detection of new links and nodes during initialization, i.e., when the node is in Init state, and their detection during normal operation, when the node is in Normal state.

The former will be referred to as initial neighbor discovery whereas the latter will be referred to as continuous neighbor discovery. While previous works [1], [2], [3] address initial neighbor discovery and continuous neighbor discovery as similar tasks, to be performed by the same scheme, we claim that different schemes are required, for the following reasons:

Initial neighbor discovery is usually performed when the sensor has no clue about the structure of its immediate surroundings. In such a case, the sensor cannot communicate with the gateway and is therefore very limited in performing its tasks. The immediate surroundings should be detected as soon as possible in

All copyrights Reserved by NCRTCST-2015, Departments of CSE  
CMR College of Engineering and Technology, Kandlakoya (V), Medchal Road, Hyderabad, India.

Published by IJECCCE (www.ijeccce.org)
order to establish a path to the gateway and contribute to the operation of the network. Hence, in this state, more extensive energy use is justified. In contrast, continuous neighbor discovery is performed when the sensor is already operational. This is a long-term process, whose optimization is crucial for increasing network lifetime. When the sensor performs continuous neighbor discovery, it is already aware of most of its immediate neighbors and can therefore perform it together with these neighbors in order to consume less energy. In contrast, initial neighbor discovery must be executed by each sensor separately.

Figure 1 shows a typical neighbor discovery protocol. In this protocol, a node becomes active according to its duty cycle. Let this duty cycle be in Init state and in Normal state. We want to have. When a node becomes active, it transmits periodical HELLO messages and listens for similar messages from possible neighbors. A node that receives a HELLO message immediately responds and the two nodes

![Figure 1.1: transmission of HELLO messages in Init and Normal states](image)

some prespecified time elapses or connectivity to a prespecified number of neighbors is detected connectivity to most of the neighbors is lost.

![Figure 1.2: Continuous neighbor discovery vs. initial neighbor discovery in sensor networks](image)

To summarize, in the Init state, a node has no information about its surroundings and therefore must remain active for a relatively long time in order to detect new neighbors. In contrast, in the Normal state the node must use a more efficient scheme. Such a scheme is the subject of our study. Figure 2 summarizes this idea. When node u is in the Init state, it performs initial neighbor discovery.

After a certain time period, during which the node is expected, with high probability, to find most of its neighbors, the node moves to the Normal state, where continuous neighbor discovery is performed. A node in the Init state is also referred to in this paper as a hidden node and a node in the Normal state is referred to as a segment node.

The main idea behind the continuous neighbor discovery scheme we propose is that the task of ending a new node u is divided among all the nodes that can help v to detect u. These nodes are characterized as follows: (a) they are also neighbors of u; (b) they belong to a connected segment of nodes that have already detected each other; (c) node v also belongs to this segment. Let degs(u) be the number of these nodes. This variable indicates the in-segment degree of a hidden neighbor u. In order to take advantage of the proposed discovery scheme, node v must estimate the value of degs(u).

II. RELATED WORK

In a Wi-Fi network operating in centralized mode, a special node, called an access point, coordinates access to the shared medium. Messages are transmitted only to or from the access point. Therefore, neighbor discovery is the process of having a new node detected by the base station. Since energy consumption is not a concern for the base station, discovering new nodes is rather easy. The base station periodically broadcasts a special HELLO message to other nodes. A regular node that hears this message can initiate a registration process. The regular node can switch frequencies/channels in order to find the best HELLO message for its needs. Which message is the best might depend on the identity of the broadcasting base station, on security considerations, or on PHY layer quality (signal-to-noise ratio). Problems related to possible collisions of registration messages in such a network are addressed in [4]. Other works try to minimize neighbor discovery time by optimizing the broadcast rate of the HELLO messages [1], [5], [6], [7], [8]. The main differences between neighbor discovery in Wi-Fi and in mesh sensor Networks are that neighbor discovery in the former is performed only by the central node, for which energy consumption is not a concern. In addition, the hidden nodes are assumed to be able to hear the HELLO messages broadcast by the central node. In contrast, neighbor discovery in sensor networks is performed by every node, and hidden nodes cannot hear the HELLO messages when they sleep.

In mobile ad-hoc networks (MANETs), nodes usually do not switch to a special sleep state. Therefore, two neighboring nodes can send messages to each other whenever their physical distance allows communication. AODV [9] is a typical routing protocol for MANETs. In AODV, when a node wishes to send a message to another node, it broadcasts a special RREQ (route request)
message. This message is then broadcast by every node that hears it for the rest time. The same message is used for connectivity management, as part of an established route maintenance procedure, aside from which there is no special neighbor discovery protocol. Minimizing energy consumption is an important target design in Bluetooth [10]. As in Wi-Fi, the process of neighbor discovery in Bluetooth is also asymmetric. A node that wants to be discovered switches to an inquiry scan mode, whereas a node that wants to discover its neighbors enters the inquiry mode. In the inquiry scan mode, the node listens for a certain period on each of the 32 frequencies dedicated to neighbor discovery, while the discovering node passes through these frequencies one by one and broadcasts HELLO in each of them. This process is considered to be energy consuming and slow. A symmetric neighbor discovery scheme for Bluetooth is proposed in [11]. The idea is to allow each node to switch between the inquiry scan mode and the inquiry mode.

The Disco algorithm is proposed for scheduling the wake-up times of two nodes that wish to find each other. For this algorithm, each node chooses a prime number; the choice depends on the required discovery time. Using the Chinese Remainder Theorem, it is proved that the wake-up periods of the nodes will overlap within the required time. However, [13] does not discuss the problem of many sensors in the same segment collaborating to reduce the energy they expend for discovering hidden nodes.

A novel low-power listening (LPL) technique, proposed in [14] to overcome sensor synchronization problems, is implemented by the B-MAC protocol [15]. The transmission of a packet is preceded by a special preamble. This preamble is long enough to be discovered if each node performs periodic channel sampling. However, this technique can usually not be used for initial neighbor discovery, and cannot be used at all for continuous neighbor discovery, because it actually requires the node to stay awake during the entire time it is searching for a new neighbor.

III. A BASIC SCHEME AND PROBLEM DEFINITION

In the following discussion, two nodes are said to be neighboring nodes if they have direct wireless connectivity. We assume that all nodes have the same transmission range, which means that connectivity is always bidirectional. During some parts of our analysis, we also assume that the network is a unit disk graph; namely, any pair of nodes that are within transmission range are neighboring nodes. Two nodes are said to be directly connected if they have discovered each other and are aware of each other’s wake-up times. Two nodes are said to be connected if there is a path of directly connected nodes between them. A set of connected nodes is referred to as a segment. Consider a pair of neighboring nodes that belong to the same segment but are not aware that they have direct wireless connectivity. See, for example, nodes a and c in Figure 4(a). These two nodes can learn about their hidden wireless link using the following simple scheme, which uses two message types: (a) SYNC messages for synchronization between all segment nodes, transmitted over known wireless links; (b) HELLO messages for detecting new neighbors.

Scheme 1 (detecting all hidden links inside a segment):

This scheme is invoked when a new node is discovered by one of the segment nodes. The discovering node issues a special SYNC message to all segment members, asking them to wake up and periodically broadcast a bunch of HELLO messages. This SYNC message is distributed over the already known wireless links of the segment. Thus, it is guaranteed to be received by every segment node. By having all the nodes wake up “almost at the same time” for a short period, we can ensure that every wireless link between the segment’s members will be detected.

To better understand the best of Scheme 1, we now compare its performance to the performance of a trivial algorithm where every node discovers its hidden neighbors independently. When Scheme 1 is used, a hidden node is discovered by all of its in-segment neighbors as soon as it is discovered by the rest of them. In contrast, when Scheme 1 is not used, the hidden node is discovered by all of its in-segment neighbors only when it is discovered by the last of them. To analyze the time slots at which these nodes are discovered,

**Scheme 2 (detecting a hidden link outside a segment):**

Node u wakes up randomly, every T (u) seconds on the average, for an x period of time H. During this time it broadcasts several HELLO messages, and listens for possible HELLO messages sent by new neighbors. The value of T (u) is as follows:

\[ T(u) = T_1 \text{, if node } u \text{ is in the Init state of Figure 2.} \]

\[ T(u) = T_N(u), \text{if node } u \text{ is in the Normal state of Figure 2, where } T_N(u) \text{ is computed according to the scheme presented.} \]

A random wake-up approach is used to minimize the possibility of repeating collisions between the HELLO.
messages of nodes in the same segment. Theoretically, another scheme may be used, where segment nodes coordinate their wake-up periods to prevent collisions and speed up the discovery of hidden nodes. However, finding an efficient time division is equivalent to the well-known node coloring problem, which is

![Fig. 3.2 Segments with hidden nodes and links](image)

A known link _______
An unknown link -----------------

IV. ESTIMATING THE IN-SEGMENT DEGREE OF A HIDDEN NEIGHBOR

To determine the discovery load to be imposed on every segment node, namely, how often such a node should become active and send HELLO messages, we need to estimate the number of in-segment neighbors of every hidden node, denoted by $\text{deg}_u(v)$. In this section we present methods that can be used by node $v$ in the Normal (continuous neighbor discovery) state to estimate this value. Node $u$ is assumed to not yet be connected to the segment, and it is in the Init (initial neighbor discovery) state. Three methods are presented:

1. Node $v$ measures the average in-segment degree of the segment's nodes, and uses this number as an estimate of the in-segment degree of $u$. The average in-segment degree of the segment's nodes can be calculated by the segment leader.

2. To this end, it gets from every node in the segment a message indicating the in-segment degree of the sending node, which is known due to Scheme 1. We assume that the segment size is big enough for the received value to be considered equal to the expected number of neighbors of every node.

3. Node $v$ discovers, using Scheme 1, the number of its in-segment neighbors, $\text{deg}_v(v)$, and views this number as an estimate of $\text{deg}_u(v)$. This approach is expected to yield better results than the previous one when the degrees of neighboring nodes are strongly correlated.

4. Node $v$ uses the average in-segment degree of its segments nodes and its own in-segment degree $\text{deg}_v(v)$ to estimate the number of node $u$'s neighbors. This approach is expected to yield the best results if the correlation between the in-segment degrees of neighboring nodes is known. An interesting special case is when the in-segment nodes are uniformly distributed.

Let $X$ be a random variable that indicates the degree $\text{deg}_u(v)$ of $v$, a uniform randomly chosen node in the segment $S$. Let $Y$ be a random variable that indicates the degree $\text{deg}_u(u)$ of $u$, a uniform randomly chosen hidden neighbor of $v$, which we want to estimate. We use the mean square error measure ($\text{MSE}$) to decide how good an estimate is. The $\text{MSE}$ is defined as $E((Y - \mu)^2)$. Since $v$ and $u$ are two random neighbors in the same graph, $X$ and $Y$ have the same distribution. Let us denote the correlation between $X$ and $Y$, $\text{corr}(X, Y)$, by $C$. Throughout the section we assume that $\text{deg}_u(v)$ is small compared to the network size. Thus, for the rest method the following holds:

$$\text{MSE}_1 = E((Y - \mu)^2) = E(Y^2) = \text{Var}(Y);$$

For the second method, we have $Y^0 = X$. Hence,

$$\text{MSE}_2 = E((Y + X - \mu)^2) = E(Y^2 + X^2 + 2XY);$$

$$= (Y^2 + 2XY + X^2)P(X = x; Y = y)$$

$$= E(X^2) + E(Y^2) + 2E(XY);$$

Since $X$ and $Y$ have similar distribution, Clearly,

$$E(X) = E(Y) = \text{corr}(X; Y) \text{Var}(X) = C \text{Var}(X): \quad (5)$$

$$\text{cov}(X; Y)$$

<table>
<thead>
<tr>
<th>Method</th>
<th>$\text{deg}_u(v)$</th>
<th>$\text{MSE}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$$</td>
<td>$\text{Var}(X)$</td>
</tr>
<tr>
<td>2</td>
<td>$X$</td>
<td>$(2 - 2C^2) \text{Var}(X)$</td>
</tr>
<tr>
<td>3</td>
<td>$CX + (1 - C)$</td>
<td>$(1 - C) \text{Var}(X)$</td>
</tr>
</tbody>
</table>

Fig. 4.1. Values obtained in tabular form

We conclude that $C \text{Var}(X) = \text{Var}(X) = C$ and $= (1 - C)$ are the values that minimize the $\text{MSE}$. Therefore, for the third estimation method, $\text{deg}_u(v)$ is estimated.

V. DATA TRANSMISSION BETWEEN NODES

This is referred to as Continuous neighbor Discovery. Now we have to discuss about how the nodes are being discovered by Continuous neighbor Discovery algorithm. At first when the sensor nodes are in Init state, we consider that all the nodes present in a network are active. Now this sensor node will search for any other sensor node which is active. If any sensor node is active at that period, the first sensor node repeatedly transmits data to the next active sensor node. The other sensor node replies back by sending the ACK packet to the previous sensor nodes and therefore the two ways communication between the sensor nodes is being established.
The sensor nodes have to be detected by the other nodes by using the joint task of all the nodes. If there is a sensor node A and a sensor node C, then the neighbors of A can be detected by B with the help of H and D. Here A is a source from where we are trying to send the data to the destination node C. To do this there exist two paths, that is AHBDC and the path ABDC. This is the initial discovery; it will show only the route from source to destination. Here orange color nodes represent the nodes that are away from the network or the dead nodes. And gray color nodes are the active nodes.

VI. METHODOLOGY

In this section, we introduce the asynchronous neighbor discovery problem, provide its corresponding theoretical formulation, and develop relevant evaluation metrics for mobile sensor networks. For the benefit of the reader, we first introduce the simple neighbor discovery problem, and later generalize the formulation to describe the continuous neighbor discovery problem. Mobile nodes in sensor networks continually move in space. Due to such motion, the network topology changes over time, nodes within communication range may move farther apart, and nodes outside the communication range move closer to each other. The neighbor discovery problem is one in which each mobile node keeps track of all other nodes within its communication range. This information changes over time and a neighbor discovery protocol is used to continuously update it. The asynchronous neighbor discovery problem is often encountered in mobile networks, where the mobile nodes discovering each other are not necessarily synchronized with each other. Here we are using two main discovery algorithms namely first one is initial discovery and second is continuous discovery. Those are explained as below.

Initial neighbor discovery

Initial neighbor discovery is usually performed when the sensor has no clue about the structure of its immediate surroundings. In such a case, the sensor cannot communicate with the gateway and is therefore very limited in performing its tasks. The immediate surroundings should be detected as soon as possible in order to establish a path to the gateway and contribute to the operation of the network. Hence, in this state, more extensive energy use is justified. In contrast, continuous neighbor discovery is performed when the sensor is already operational. This is a long-term process, whose optimization is crucial for increasing network lifetime.

Continuous neighbor discovery

When the sensor performs continuous neighbor discovery, it is already aware of most of its immediate neighbors and can therefore perform it together with these neighbors in order to consume less energy. In contrast, initial neighbor discovery must be executed by each sensor separately. In a continuous neighbor discovery we are detecting all of its neighboring nodes as well as selecting the shortest path for transferring the file or the data.

VII. IMPLEMENTATION

Modules

A. Client-Servers
B. Detecting all hidden links Inside and outside the segment
C. Neighbor Discovery Model

A. Client – Server

Client-Server computing is distributed access. Server accepts requests for data from client and returns the result to the client. By separating data from the computation processing, the compute server’s processing capabilities can be optimized. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.

B. Hidden link participate inside and outside the segment

This scheme is invoked when a new node is discovered by one of the segment nodes. The discovering node issues a special SYNC message to all segment members, asking them to wake up and periodically broadcast a bunch of messages. This SYNC message is distributed over the already known wireless links of the segment. Thus, it is guaranteed to be received by every segment node. By having all the nodes wake up almost at the same time. For a short period, we can ensure that every wireless link between the segment’s members will be detected. A random wake-up approach is used to minimize the possibility of repeating collisions between the messages of nodes in the same segment. Theoretically, another scheme may be used, where segment nodes coordinate their wake-up periods to prevent collisions and speed up the discovery of hidden nodes. Since the time period during which every node wakes up is very short, and the message transmission time is even shorter, the probability that two neighboring nodes will be active at the same time.
C. Neighbor Discovery Model

Neighbor Discovery is studied for general ad-hoc wireless networks. A node decides randomly when to initiate the transmission of a message. If its message does not collide with another message, the node is considered to be discovered. The goal is to determine the message transmission frequency, and the duration of the neighbor discovery process.

Initial Discovery Algorithm

INPUT: Number of nodes, source IP address, destination IP address.
OUTPUT: Discovers the path from source to destination.
1. Enter the number of nodes and give the destination IP address.
2. Insert the source node into the discovery table database.
3. Discover the nearest node to the source node and enter it into the discovery table, for next operations consider it as the source node.
4. Find the nearest node to previously found neighbor.
5. Repeat the above steps until all nodes are found.
6. Discovery table is now having the entry of nearest node to source and destination. Using these entries the data is transferred from source to destination.

Continuous Neighbor Discovery Algorithm

INPUT: Number of nodes, source IP address, destination IP address.
OUTPUT: Discovers the efficient path from source to destination.
1. Enter the number of nodes and give the destination IP address.
2. Insert the source node into the discovery table database.
3. Discover the nearest node to the source node and enter it into the discovery table, for next operations consider it as the source node.
4. If node exists go to step 3.
5. Discovery table is now having the entry of nearest node to source and destination. Using these entries the data is transferred from source to destination.
6. Discovery table is the database maintained to keep track of the neighboring nodes.

VIII. COMPARISON

The already existing system which we have referred, in that they have considered the sensor network as a static network. The number of nodes in that are fixed and the positions are also fixed. They have given the energy for each node, after some time when the energy completes then that node is considered as a dead node and data is transferred through remaining active nodes. After some amount of time all the nodes will die, and there is no network is present to transfer the data. But in real world application the sensor network keeps changing, means nodes may added or deleted from the network. Hence we are considering the network as a dynamic sensor network.

Where we are considering the number of nodes and positions of the nodes are keeps on changing regularly.

We consider the following criteria:
- Number of nodes may increases in sensor network.
- Number of nodes may decreases in sensor network.
- Energy level is fixed.
- The following are the disadvantages of existing system
  - No clues about structure of immediate surroundings.
  - Heavy traffic.
  - Long term process.
  - Greater expense of energy

To overcome of this problems we are introduce the following
- Removing dead nodes is avoided
- Calculated service time and bandwidth.
- Traffic is controlled and the process is smoother
- Traffic is controlled and the process is smoother.
- Efficient use of energy and time

IX. CONCLUSION

We exposed a new problem in wireless sensor networks, referred to as ongoing continuous neighbor discovery. We argue that continuous neighbor discovery is crucial even if the sensor nodes are static. If the nodes in a connected segment work together on this task, hidden nodes are guaranteed to be detected within a certain probability P and a certain time period T , with reduced expended on the detection.

We showed that our scheme works well if every node connected to a segment estimates the in-segment degree of its possible hidden neighbors. We then presented a continuous neighbor discovery algorithm that determines the frequency with which every node enters the message period. We simulated a sensor network to analyze our algorithms and showed that when the hidden nodes are uniformly distributed in the area.

REFERENCES