A self-organizing routing algorithm for wireless sensor networks

This item was submitted to Loughborough University's Institutional Repository by the/an author.


Additional Information:

- This is a conference paper [© IEEE]. It is also available at: http://ieeexplore.ieee.org/ Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Metadata Record: https://dspace.lboro.ac.uk/2134/4132

Version: Published

Publisher: © IEEE

Please cite the published version.
This item was submitted to Loughborough's Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
Abstract—Wireless sensor networks (WSN) are designed to collect and process sensory data from environments. Some environments are dangerous or un-reachable to human beings and it is difficult to replace sensor nodes when they are out of battery or even destroyed, i.e. wireless sensor nodes are in general prone to failure. This kind of characteristics require WSN to detect whether or not its next destination is still available (alive) and to maintain a transferring path if the next destination in the route does not exist (dead). In the normal state, nodes are in power-saving ‘sleep’ state. When a route is created for some purpose, all nodes in this route will be active and be ready to respond requests from its neighbors. Our approach is to maintain the routing table up-to-date by sending message from a last node to its next node and judging whether the next node is alive according to the response. If problems happen, node will self-organize and try to maintain transferring.

I. INTRODUCTION

Wireless sensor network (WSN) provides an economical, effective and fully distributed way to collect and process data from environments. As the next generation of network, WSN can be used in building, industrial, home, and transportation systems automation [1]. The benefit brought with WSN is that it has ability to remotely monitor non-easily accessible areas [2]. Sensory data comes from multiple sensors organized as certain topology in distributed locations. As a future direction of network, WSN enhances people’s ability to communicate with nature rather than only receive logical data from conventional network.

Normally sensor nodes are deployed outside. This characteristic provides challenges to the design of sensors, including safety, noise in radio communication, energy consume. As mentioned above, most environments in which WSN are applied are not accessible and dangerous to people, like space, original forest, or shipboard. These nodes are difficult to be diagnosed or replaced after deployment. When they encounter problems or stop work, WSN should be able to remedy error and re-organize network to maintain reliability, especially in vital mission such as fire fighting, and martial action. Reliability is one of the hottest topics in the development of Wireless Sensor Networks. An unreliable network can not be used to serve users. Applications based on WSN ask for accurate, integrity, and availability of data, and then a practical WSN must be provided with guarantee of reliability. The point is that sensor networks have different constraints compared with traditional wired networks [3]. First, energy consumption should be considered. Sensor will not be recharged during its lifetime and the current level of battery is still not satisfying needs. Any wasted action will shorten sensors’ useful lifetime. Second, sensor’s computing limitation does not allow complex tasks which serves for certain objectives. Third, sensors in WSN communicate by radio. Noise in the radio and strength of link will affect the rate of loss of package loss.

Generally speaking, problems of reliability will happen in two situations. First, the whole network is in “un-working” state. That means only system messages are active among nodes, like initial information, identification information, or routing detection. Exiting routing protocols have their own ways to make sure each part of networks’ is available in this situation. Second is “working” state. In this situation data are “traveling” from one node to another node according to pre-defined path created by a routing protocol. An efficient algorithm of data transferring should be provided. To avoid additional load of nodes, this algorithm can use existing resource and make the least modifications.

The rest of the paper is organized as follows: in section II, we list some existing ways provided by researchers to guarantee reliability in WSN. In sections III, IV, we present how to use “Package-Count” to improve reliability of transferring data in WSN and a discussion of simulation result. We conclude the self-Recognizing routing algorithm and future work in section V.

II. RELATED WORK

Many researchers have provided ways from different angles to guarantee reliability in the use of WSN.

2.1 Topology

Multiple sensors are used to monitor a same location at a same time. Data from neighboring nodes can be used to distinguish the correctness of local data. Redundant information can be utilized to improve reliability of data. For example, mesh network is a good choice. Some successful cases are using this topology to improve reliability of transport [4]. Figure 1 shows a topology of smart mesh network deployed in business. Data package hop from one node to another in the mesh network until it reaches its destination. If one node fails, data always can find another way to go around the failed one. Redundant nodes and dynamic routing help the mesh network to exert its advantage.

In a traditional star topology, many nodes connect to a central node and every operation is managed by the central node. It is not easy for nodes to complete self-organizing
work when a fault happens; especially when the central node fails. In a mesh network, nodes are connected to each other and this makes nodes easier to communicate with each other.

2.2 Transfer mechanism

Some researchers suggest that data package should not have to wait for route discovery completing. In most of the routing protocols, data packet is only sent after route discovery package gets response and a full path has been created. They provide Persistent Dynamic Routing, which allows data package being transferred as soon as it is ready. Precondition is that the whole network must be flooded at the beginning because of the route discovery. Data package tries to reach destination by using neighbor information and available route discovery which was collected before and cached in a single node. This way is suitable for highly dynamic network and can significantly decrease the overhead. The problem is once part of the network has been interfered, route and neighbor information will not be updated in time [5].

![Smart Mesh Network](image)

Figure 1 Smart Mesh Network allows nodes communicate with each other

2.3 Transport Layer

Traditionally the difficulty in transport layer is that the data may be larger than the network MTU (Maximum Transmission Unit). If a data packet is lost [3], the whole data entity will be useless. But in WSN, the similar problem is always caused by interference or lower power. Problems may be solved in two ways. First, sink nodes take responsibility to check the integrity. If data is lost, the sink node will ask for retransmitting the lost data. Second, each node on the path will cache part of data entity. When a sensor node identifies any data loss, it will send back a repair request to the next hop towards the source. If the requested hop does not have data, the request will be forwarded in the way toward the source until the source node is reached. Data aggregation can be associated with this hop-by-hop request. If requests are first sent to aggregation hop, it will have more chance to find out missing data and therefore time and energy are saved. An obvious disadvantage is that the design of node will be more complex. That is not good for energy saving.

III. METHODOLOGY

The purpose of this research is to design a self-recognizing routing algorithm for use in the wireless sensor network. The objective of the self-recognizing routing algorithm is to find an optional way when one node existing in active route is broken (dead). Assuming nodes on this route are transferring data and this route is created by a routing protocol before any transferring is taken place. Once one of nodes encounters any problems such as too low power to work, or being inferred, data transferring must be maintained to guarantee the quality of transferring. This algorithm will be used during the whole process of transferring in order to detect whether or not a node is still available. In some sense, it can be thought as a supplement of the exiting routing protocol.

3.1 Self-Recognizing

Designing an algorithm to detect nodes’ state and making adjustment when problems occur is a very complex task. The goal of this algorithm is to detect nodes state in order to maintain transferring path even some nodes are destroyed. Because sensor node has limited computing capacity and storage for additional tasks, the algorithm should be simple and have minimal affect on other nodes. As a supplement of the exiting routing protocol, self-recognizing should work with the existing topology and routing protocol.

3.1.1 Topology Choice

Basic topologies include Star, Ring, Bus, Tree, Fully Connected, and Mesh. Figure 2 shows topologies of each type. We choose Mesh topology because it has greater advantage than others. Star, Bus topologies are limited with reliability. If the central node or base line is broken, the network will fail. Ring and Tree topologies are not easy for nodes to communicate with each other. Fully Connected topology is too difficult to achieve. Mesh topology is flexible and easy to be re-organized. The whole network will not be affected even a small number of nodes fail at the same time. When one node in the route encounters problems, a sender can easily find an optional way around the failed node.

3.1.2 Routing Protocol

Self-Recognizing algorithm works when data is being transferred. A routing protocol should be employed to create a route first. And this routing protocol can provide a foundation for the self-recognizing algorithm. We choose Ad hoc On Demand Distance Vector (AODV) in this study.
AODV is a routing protocol designed for ad hoc mobile networks. It can build routes between nodes. When a source node desirers a route to a destination, it broadcasts a route request (RREQ) packet across the network. Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node. If nodes do not have any route to the destination, they will forward the packet. If they have a route to the destination or they are the destination, they will send a route reply (RREP) to the source node which is transferred back along the pointers backwards from the destination in the route table. When nodes receive a reply packet, they set up forward pointers to the destination instead of backward pointers in route tables. After receiving a reply packet, the source node will choose the shortest route and continually send data packets. Then data packets can track the pointer to the destination in each node. There is a route table as shown in Table 1.

Table 1: Route table in nodes. Dest means destination, NextHop means the next hop which can point to the destination from current node. numHops means how many hops are requested from the current node to the destination.

<table>
<thead>
<tr>
<th>Dest</th>
<th>10</th>
<th>32</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>NextHop</td>
<td>3</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>numHops</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

3.2 Self-Recognizing Algorithm

Self-Recognizing Algorithm is designed to work when data is transferred. Suppose that WSN has chosen Mesh as the network topology and the AODV protocol as the routing protocol used in the mesh network. When data are transferred, they do not know whether the next node is still available or not. They are forwarded again and again to the next nodes until reaching the destination. If one or more nodes failed on the route paths, the transferring will not stop and data packets are lost. To detect the availability of route’s and minimize the nodes load, we modify the form of data packet as follows. A common form of data packet is shown in Figure 3.

![Figure 3. Form of data packet. Header includes control information of data packet. Seq means the sequence number of the current packet. TTL is time of living of data packet and Data is content of packet. Src is the address of source data. Dest is the destination of data. App is the current application. Length is the length of packet.](image)

When a node receives a packet, it will check the header of packet to decide how to deal with this packet. We add a new item “Type” in option “Header”, shown in Figure 4.

![Figure 4. Form of data packet with new item “Type”](image)

The value of “Type” will be 0 or 1. If ‘Type’ is 0, packet will be treated as a common data packet, otherwise, the node which receives this packet should send back a state response to its last node to report its availability. The process is shown in Figure 5.

![Figure 5. Node 1 sends data packets node 3, via node 2. ■ Type 1 data packet. □ Type 0 data packet. ◆ Response packet, sent when node receives ‘Type’ 1 data packet.](image)

We add a counter in the route table for each currently used “NextHop”. Table 2 shows the route table modified.
Table 2: Modified route table which has a counter for each next hop.

<table>
<thead>
<tr>
<th>Dest</th>
<th>10</th>
<th>32</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>NextHop</td>
<td>3</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>Counter</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>numHops</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Every time the node sends/forwards a Type 1 data packet to a next hop, the relevant counter will be added 1. If the node receives a response message from the next hop, the relevant counter will be reset to 0. Before sending the packet 1, the node should check the counter first. If it has reached or exceed a pre-defined number, that means the next hop may encounter problems. Considering about time delay or congestion in the network, the counter may be increased to another pre-defined number which is greater than the first pre-defined number. If the second pre-defined number is reached, the next node will be thought being unavailable. Whatever happens to the next node, it can not be used to transfer data any more. Because each node in this route should be active during all the process of transferring, this method of judgment is reliable. After knowing the next node is unavailable, the current node will notice the source node to stop sending data and generate a new route before sending any data packet again.

3.3 Data Recovery
Finding a new route is necessary when transferring is interfered. But the recovery has two solutions. One is about the data integrity. The data can be no use without any part of data packet. In this situation, the routing protocol should try to recovery packet accurately. Each packet lost should be resent again. Another is about the packets' usefulness. For purpose of analysis, continuous data might be required. For example; researchers want to analysis 100 data. If data 50 is lost, then data 1-49 and 51-101 still can satisfy needs. Trying to find the 50th data is neither necessary nor important. Especially for a mini sensor, if time has gone and there is no enough space to restore previous data, re-sending the 50th data is not meaningful. Using “counter” which records special data packet with ‘Type’ 1 can work out how many data packets have been lost under these two conditions. Actually this “counter” should work with the application layer in networking software. If the reliability of network only requires a certain number of data, applications can calculate the number of packets which have been lost simply and ask the source node to supplement. If it requires a high degree of integrity, applications can do a complex task based on “counter” to recover the lost data packets. Those lost data packet may retrieve by some other methods, like data aggregation in certain nodes. Mesh network is a good choice for this. In this simulation, we only try to recover a certain number of lost data packets.

IV. IMPLEMENTATION

4.1 Simulator
TinyOS is an event-driven operating system intended for sensor networks with limited resource. It is designed by researchers in UC Berkeley. As an operating system of WSN, TinyOS meets basic requirements of sensor nodes. It features a component-based architecture which enables rapid innovation and implementation and minimizes code size to make constraint memory of sensor node work effectively. The TinyOS system, libraries and applications are written in nedC, a new language for programming embedded system. nesC is a version of C which has a C-like syntax. It supports the TinyOS concurrency model. In nedC, programs consists of components which are wired together to execute an application. These components are assembled by an interface, which is the only point of access to the components and is bi-directional. nesC also defines a concurrency model, based on tasks and hardware event handlers, and detects data races at compile time. The principle of nesC is to allow application designers to build components that can be easily composed into complete, concurrent systems, and yet perform extensive checking at compile time. [7]

4.2 TOSSIM
TOSSIM is a discrete event simulator for TinyOS sensor network. Before compiling a TinyOS application for a mote, users can compile it into TOSSIM, which runs on a PC. TOSSIM builds directly from TinyOS code. This make users debug, test and analyze algorithms in a controlled environment easily. TOSSIM simulates the TinyOS network stack at the bit level, allows experimentation with low-level protocols in addition to top-level application systems. It simulates each individual ADC capture and every interrupting in the system. However, though TOSSIM captures TinyOS behavior at a very low level, code which runs in a simulation might not run on a real mote. Although TOSSIM times interrupts accurately, it does not model execution time. TOSSIM also does not model radio propagation and power draw or energy consumption. Instead, it provides radio abstraction of independent bit errors between nodes and can be adjusted to add annotations to components that consume power to provide information about power states.

4.3 TinyVIZ
TinyVIZ is a Java visualization and actuation environment for TOSSIM. It allows TOSSIM simulation to have an extensible graphical user interface for debugging, visualizing and interaction. The TinyVIZ engine uses an event-driven model, which allows easy mapping between TinyOS’s event-based execution and event-driven GUIs. Using a simple plug-in model, users can develop their own function to interact with simulation [8].

3391
4.4 Simulation

4.4.1 Scenario I
Assume in a mesh wireless sensor network there are 30 nodes active. Node 0 is the sink node. Node 26 is the source node. Node 26 wants to send data to the sink node. After the work of AODV protocol, node 26 generates a route to send data continuously to the sink node. The route is 26→1→14→10→0. Simulation result is shown in Figure 6.

![Figure 6. Source node 26 sends data packets to sink node 0. The route is 26→1→14→10→0.](image)

4.4.2 Scenario II
When node 1 is disabled, the route is destroyed and transferring from node 1 to the sink node stops. It is shown in Figure 7.

![Figure 7. Node 1 is disabled. Source 26 still sends data to node 1 continuously, but transferring from node 1 to the sink node stops.](image)

4.4.3 Scenario III
Source node 26 has not received responses from node 1 for a long time and the “counter” in its route table for node 1 has reached the “pre-defined number”, it stops transferring, generates a new route 26→2→14→10→0, and starts to send data packet again. It is shown in Figure 8.

![Figure 8. Source node 26 generates a new route to send data packet. The new route is 26→2→14→10→0.](image)

Table 3: Result of packet transferring

<table>
<thead>
<tr>
<th>Transfer Model</th>
<th>Packet</th>
<th>Lost Packet</th>
<th>Lost Rate</th>
<th>Recovery Packet</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal (Without interrupting)</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>Transfer with interrupting</td>
<td>100</td>
<td>13</td>
<td>13%</td>
<td>0</td>
<td>87%</td>
</tr>
<tr>
<td>Transfer with interrupting and recovery</td>
<td>100</td>
<td>13</td>
<td>13%</td>
<td>12</td>
<td>99%</td>
</tr>
</tbody>
</table>

Result of packet transferring is shown in Table 3. 100 packets are sent from the source node to the sink node. If there is no interrupting, packet loss is 0 and success rate is 100%. If transferring was interrupted, 13 packets were lost and success rate is 87%. If “counter” was used to calculate the number of lost data packet, the source node can re-send data packets in terms of the result of calculation. 12 packets can be recovered and success rate is 99%.

We set a “type 1” data packet every 3 packets. Under normal situation, the “counter” in the route table will be reset immediately after sending “type 1” data packet. That is because the next node always responds in time. When node 1 is unavailable, it will not respond to node 26 and the “counter” increases. If the “counter” equals 2, node 26 judges that there may be something wrong with node 1. When the “counter” equals 4, the source node 26 stops transferring data packets to node 1 and tries to generate a
new route. After this work is done, node 26 resends lost data packets calculated by the “counter” and reset “counter” before recovering transferring.

![Graph showing success rate vs. number of packets]

- Normal
- Transfer with interfering and recovery
- Transfer with interfering

Fig 9. Comparison of number of data packets versus success rate

More simulations have been carried out. Fig. 9 shows the performance of self-recognizing mechanism. With the increase of the number of total packets, self-recognizing mechanism works better. It is because the source stops sending out packets before lost packets are found.

4.5 Discussion

Advantage: Package-counter makes use of exiting routing and route information to detect nodes’ states. It can save storage space without additional overhead. Especially it can provide useful information for retransferring. The self-recognizing mechanism can work with applications to improve the reliability of WSN.

Disadvantage: Routing protocols used in WSN have their own mechanisms to provide reliability. Time interval is the most common way to judge whether transferring is successful or not. Package-count can be used in simulation study, but how well it associates with real routing protocol is still unknown. In a large range of WSN, frequent counting for counter may lead to congestion.

V. CONCLUSION AND FUTURE WORK

To improve the self-recognizing mechanism, two improvements must be considered. First, how often do we set up a “type 1” data packets and how to choose the pre-defined number. This result should associate with various factors. Signal strength, noise level, response time, congestion level will be taken into consideration. Second, how to generate a new route is worth of discussing. In simulation, when the current node finds out problems with the next node, it “tells” source node to look for a new route. Due to the equality of nodes, it is possible for the current node to generate a new route rather than noticing the source node. This may save time and energy.

Whatever routing protocol is used in WSN, basic communication still takes place between individual nodes. Routing protocol can check network’s integrality when it is in idle time. But if nodes are busy with transferring data, they may not respond the “check” message requested in time. Many ways have been provided to assure reliability, trying best to get much better scalability and throughput of wireless sensor networks without sacrificing reliability is the common purpose. Also, energy and space saving must be considered as an important point in the design of WSN.

REFERENCES


3393