A Software Update Method using Clustering WSNs

Hyeyeong Jeong and Byoungchul Ahn
Dept. of Computer Engineering, Yeungnam University, Gyungsan, Korea

Abstract: Wireless Sensor Networks (WSNs) are applied to many monitoring applications. Present sensor nodes can perform many functions at the same time and contain complex software. During the lifetime of sensor nodes, they are required to reprogram their software because of their new functions or software bug fixes. Since typically sensor nodes are inaccessible physically or it is very difficult to upgrade their software by one by one, it is necessary to upgrade software of sensor nodes in WSNs remotely. This paper presents an energy efficient method by selecting an optimal relay node. The Cluster Head Relay (CHR) method is compared with SPIN and RANDOM method. Three methods are simulated using NS-2 with the same environmental parameters. Simulation results show that CHR shows that the update time is reduced up to 17% and the number of relay nodes is reduced up to 19% compared with other two methods.

Keywords: firmware upgrade, clustering, SPIN, relay node, WSN

1 Introduction

WSNs are becoming one of the most recent Ubiquitous computing technologies. WSN applications are rapidly spreading since recent developments of semiconductor technology gives opportunity to implement many functions in a sensor node. These sensor nodes can be installed to observe inaccessible environment, traffic monitoring and disaster spread places.

After deploying sensor nodes, their battery is not replaceable or rechargeable since they are installed on areas where are difficult to access. Therefore, efficient energy consumption methods are important in WSN applications. Most energy is consumed by radio transmission modules in sensor nodes. Energy consumption of radio modules can be saved by shortening of the transmission distance. In order to reduce the transmission distance, relay nodes have to be chosen optimally. In this paper, we present a new method to update firmware in WSNs, by selecting optimal relay nodes to increase energy efficiency.

In Chapter 2, related work for firmware update is given. Chapter 3 represents the network model and assumptions of proposed method and Chapter 4 describes, the proposed method to select relay nodes, CHR algorithm, is described. In Chapter 5, simulation results of the proposed work are discussed. Conclusion is discussed in chapter 6.

2 Related Work

Stephen et al. suggest a model to upgrade software in WSNs[8][9]. This model presents the theoretical approaches to upgrade software. They have not presented any simulation results or experiments to verify their model. They validate their model against three different systems, representing three classes of software update: static/monolithic updating(MOAP), dynamic/mobile agent-based updating (Mate) and dynamic/component-based updating(Impala).

Infuse implements selective retransmission Go-Back-N scheme using TDMA protocol’s implicit acknowledgement and back pressure mechanisms to provide reliable transfer. In order to save energy, TDMA scheme’s listening method is used. Main feature of Infuse method is continuously sends data to the next node from a predecessor node[1].

Since control software contains execution code for a processor of sensor nodes, it is very important to maintain reliable data transfer. A method for reliable data transfer in WSNs is developed for 1:1 communication such as S-TCP and RMTS[3][4]. But 1:1 communication methods are inefficient to upgrade many nodes on WSNs. If these methods are used for re-programing sensor nodes of WSNs, each node must be updated first and retransmit the software upgrade data to another node one by one. Therefore it is necessary to develop an efficient upgrade method for sensor nodes with fast upgrade time and small data retransmission.

The direction to upgrade control software is the opposite direction of normal data transfer[5][6]. It is necessary to study for large data transfer from one node to many nodes efficiently. There are some researches about software upgrades for sensor nodes. But they are focused on system management, not an upgrade itself [7]. In this paper, an energy efficient software upgrade method is described by comparing the other methods.

3 Proposed Upgrade Model

All sensor nodes of WSNs are assumed to use the same hardware configuration such as the same memory size and the same processor and so on. It means that all sensor nodes use the same software version. And a distance between two nodes is the same and the location of nodes is fixed.
In this paper, some assumptions for software update model described are given as follows:

1. Wireless Sensor Network uses CSMA/CA based mesh structure ad-hoc network
2. There is only one base station in the network
3. All nodes use same firmware
4. All nodes are fixed on position

3.1 Energy Consumption Model

In this section, energy consumption model to update software is described. Total energy consumption of nodes can be represented as the sum of energy used for transmitting and receiving control data, software transfer and retransfer data in case of error.

Figure 1 shows data transmission pattern when nodes are placed by one line. In Figure 1, “r” represents node’s transmission radius and “l” defines the distance from the base node to the last node. The black node is a base node and gray nodes are relay nodes to update software. White nodes update software only without relaying software update. In Figure 1, the minimum number relay nodes, \( N \), from the base node to the last node can be calculated by Equation (1).

![Figure 1. One dimensional placement of nodes for software update](image)

\[
N = \frac{l}{r} \quad (1)
\]

where, \( r = \text{radio radius} \)
\( l = \text{the distance from the first node to the last node} \).

Nodes are placed in 2-dimensional space and they are fixed in place. The average number of receiving nodes, \( N_a \), from transmitting node can be calculated by Equation (2).

\[
N_a = \frac{\pi \cdot r^2}{A} \cdot N_i \quad (2)
\]

where, \( r = \text{Radius of the wireless nodes} \)
\( A = \text{The area of the sensor field} \)
\( N_i = \text{Total number of nodes} \).

Energy consumption for transmission and reception can be expressed as energy consumption \( (E_t) \) of a transmitting node and energy consumption \( (E_r) \) of a receiving node. Equation (3) defines the energy consumption of a receiving node. And Equation (4) expresses energy consumption of a transmitting node.

\[
E_t = (D_f \cdot e_t + D_c \cdot e_c) \quad (3)
\]

\[
E_r = (D_f \cdot e_t + D_c \cdot e_c) \quad (4)
\]

where, \( D_f = \text{number of packets for firmware} \)
\( e_t = \text{transmitting energy per unit data} \)
\( D_c = \text{the amount of Control data for error recovery} \)
\( e_c = \text{receiving energy per unit data} \).

To select a relay node in wireless sensor networks, the energy level of the relay node should be checked since software update time and energy level of each node might be different. Equation (5) calculates the energy consumption \( (E) \) of all nodes in wireless sensor networks.

\[
E = (E_t \cdot N_a + E_r) \cdot N \quad (5)
\]

where, \( N = \text{number of relay node} \).

3.2 Firmware Update Time

Time to update software can be expressed by the sum of data transmission time, error recovery time and reprogramming time at a node. Required time \( (T_{data}) \) for data transfer is calculated by Equation (6) and overall error recovery time \( (T_{err}) \) is equal to Equation (7).

\[
T_{data} = \left( \frac{P_d + P_h}{\text{bit rate}} + P_o \right) \cdot c_f \quad (6)
\]

where,
\( P_d = \text{Data packet size} \)
\( P_h = \text{Packet header size} \)
\( P_o = \text{Wireless channel access time} \)
\( C_f = \text{the number of packets for software updates} \).

\[
T_{err} = \left( \frac{P_d + P_h}{\text{bit rate}} + P_o \right) \cdot c_f \cdot P_{err} + \left( \frac{P_d + P_h}{\text{bit rate}} + P_o \right) \cdot C_f \cdot P_{err} \quad (7)
\]

where,
\( P_d = \text{Data packet size} \)
\( P_{err} = \text{Transmission error rate} \).

\[
T_{step} = T_{data} + T_{err} + T_{op} \quad (8)
\]
where, $T_{\text{op}} = \text{Update time after receiving data.}$

Update time ($T_{\text{op}}$) of node in single hop is calculated by Equation (8). Network software update time ($T$) is expressed as double of all nodes software update time. Network software update time is shown in Equation (9)

$$T = 2T_{\text{op}} \quad (9)$$

where, $l = \text{distance from base node to last node.}$

4 Relay Node Selection

4.1 Relay Node Selection

The updating time and energy consumption are affected by the selection method of a relay node. Three selection methods, which are SPIN, random selection and clustering, are considered to update software.

(1) SPIN is a routing method that utilizes metadata to reduce energy loss and data redundancy to twice [10]. Operation steps of SPIN are shown in Figure 2. In Figure 2 (a), if a source node has data to transmit, it broadcasts ADV message to inform other nodes. In Figure 2(b) a node to receive data responds with REQ message. In Figure 2(c), data is transmitted from a source node, black node, to a requested node, B.

(2) Random: A relay node is selected randomly by the relay node of the previous stage.

(3) Cluster Head Relay (CHR). At this time sensor networks are grouped as clusters. Nodes in a cluster send their collected data to their cluster head. Cluster heads collect data from nodes, and send them to a base station. Configuration of cluster heads follows LEACH’s method [11]. Among the received data, the base station searches cluster head ID and selects a cluster head as relay node. If communication between the base station and a cluster head fails, the base station selects a relay node among member nodes. However, cluster heads must have enough energy and strong signal to reduce communication error or fail during updating software.

4.2 Cluster Head Relay Node (CHR) Algorithm

In LEACH method, gathered data are sent to the base station by cluster heads. When data is transmitted to the base station, CH information is added. In Figure 3, cluster head nodes 5, 7, 9, 12 and 17 in the clusters send gathered data to the base station.

The base station searches node IDs of cluster heads among received data and selects these nodes as relay nodes. Before updating firmware, nodes to be updated acquire event data and send them to neighbor nodes. CHR algorithm is shown as Figure 4. After receiving “Relay-Start” message, software update procedure is started.
Data send node

Broadcast status information periodically
If (receive “relay-start” && exist proper neighbor node) {
    Send “data-start” to neighbor nodes.
    Send data
}
While(No. data receive nodes > 0) {
    If(receive “reprogram-done”) {
        Select one node.
        Send “relay-start” to selected node.
        Decrease No. of data receive node.
    }
}
Go to 1.1

Data receive node

Broadcast status information periodically
If (receive “data-start”) {
    If(receive firmware ver. > stored firmware ver.) {
        Receive data and store it to memory
        Request missing or lost packet
        Reprogramming it-self.
        Send “reprogram-done” to data send node.
    }
}
If receive “relay-start”) {
    Go to 1.1
}Else {
    Go to 2.1
}

Figure 4. Proposed CHR algorithm

Relay nodes broadcast “Data-Start” message and new software version to their neighbor nodes, and then inform software updating transmission is started. After software upgrading is finished, updated nodes send “Reprogram-Done” message to their relay nodes.

5 Simulation and analysis of results

5.1 Simulation Environment and Evaluation

As a simulation tool, NS-2 is used. There are one base station and 99 sensor nodes in the network and they are distributed uniformly. CSMA/CA 802.11 and 802.15.4 are used as the wireless protocol. Table 1 shows other simulation parameters.

Table 1. Simulation Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC Protocol</td>
<td>CSMA/CA</td>
</tr>
<tr>
<td></td>
<td>Back-off Window 2–26</td>
</tr>
<tr>
<td></td>
<td>Slot Time=0.384ms</td>
</tr>
<tr>
<td></td>
<td>IFS=1.664ms</td>
</tr>
<tr>
<td>RF Transmission Range</td>
<td>60m</td>
</tr>
<tr>
<td>Wireless Bandwidth</td>
<td>250Kbps</td>
</tr>
<tr>
<td>Number of node</td>
<td>100</td>
</tr>
<tr>
<td>Transmit / receive energy consumption</td>
<td>75.9mW(TX)/62.7mW(RX)</td>
</tr>
<tr>
<td>The distance between nodes</td>
<td>20m–50m</td>
</tr>
<tr>
<td>The size of the packet data</td>
<td>64byte ~ 256 byte</td>
</tr>
<tr>
<td>Data Header Size</td>
<td>8 byte</td>
</tr>
<tr>
<td>Protocol overhead (IP+MAC)</td>
<td>16byte(8byte +8 byte)</td>
</tr>
<tr>
<td>Firmware size</td>
<td>128K byte</td>
</tr>
<tr>
<td>The data transfer rate</td>
<td>Wireless Bandwidth 70%</td>
</tr>
</tbody>
</table>

To compare the performance of software update for three methods in Section 4, simulations are taken to analyze four factors below.

① Energy Consumption
② All node update time
③ Number of relay node
④ Loss and error data

5.2 Simulation Results

Distance of nodes is measured at 20m, 40m and 50m. Energy consumption, update time, the number of relay nodes and data loss and error are measured. When the distance between nodes is at 40 meter, simulation results of power dissipation are very close to expected theoretical results. In this paper, the node-to-node distance of 40m is used for simulations.

Figure 5 shows the total energy consumption to update software for all nodes. The total energy consumption of wireless sensor networks is calculated by Equation (5). When the packet size is 256 bytes, the energy consumption of nodes is equal to 101 joules. When the packet sizes are 64 bytes and 92bytes, energy consumption is increased for three methods, because smaller packet sizes generate more collisions. When data packet size changed from 92 bytes to 224 bytes, energy consumption of the CHR method decreases from 115 Joules to 105 Joules.
The total software update time is calculated by Equation (9). When the packet size is 256 bytes, the update time is 120 seconds. When the packet size is 64 bytes, the software update time takes from 190 seconds to 240 seconds. Although the packet size is between 96 bytes and 224 bytes, the software update time is between 105 and 135 seconds.

Figure 7 shows the number of relay nodes for three methods. CHR shows the best performance of the three methods. When the packet size is 256 bytes, the number of relay nodes is the lowest number as 35 nodes.

Figure 8 shows the data error rate of the software update. When the packet size is 256 bytes, the data loss rate is 1.05 bytes. During the packet size is taken as 64 bytes, the rate of data loss is high because of data collisions. When the packet size is between 128 and 256 bytes, data error rate is lowest.

Figure 8 shows the data error rate.

From the simulation results, CHR method shows better performance than SPIN and RANDOM. To select relay nodes, the energy status of relay nodes should be considered. The advantage of CHR method shows less energy consumption than other methods.

Also one disadvantage of SPIN is that data transfer is not guaranteed. For example, when a destination node needs to update its software, it could be located since it is far from a source node. The relay node does not send “REQ message” to the source node and does not broadcast to its neighbor nodes. As a result, the destination node cannot update its software. Therefore, the destination node needs to find another relay node and dissipates more energy. RANDOM method selects relay nodes randomly and update time and energy consumption can be increased by data collisions.

6 Conclusion

In this paper, three software update methods are compared using NS2 simulation tool. When node-to-node distance is at 40 meters and the packet size is between 128 bytes and 256 bytes, it shows best result. CHR method decreases the number of relay nodes to 12%~15% and its data error rate is 0.5%~1.5% compared with other methods. Energy consumption is reduced to 0.07~19% and update time is reduced to 0.22~0.25% compared with other methods. When the packet size is changed from 64 bytes to 256 bytes, the packet size of 160 bytes performs better performance compared with other packet sizes, which are 64 or 128 bytes respectively. Please address any questions of this paper to Byoungchul Ahn by Email (b.ahn@yu.ac.kr).
7 Acknowledgement

This work (Grants No. C0149241) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2013.

8 References


