Lightweight Reprogramming and Energy Balancing in Wireless Sensor Networks

Dinh-Sy Do and Younghan Kim

School of Electronic Engineering Soongsil University, Sangdo-Dong, Dongjak-Gu, Seoul 156-743, Republic of Korea

Correspondence should be addressed to Younghan Kim; younghak@ssu.ac.kr

Received 9 January 2015; Accepted 30 March 2015

Academic Editor: Fan Wu

Copyright © 2015 D.-S. Do and Y. Kim. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Reprogramming in wireless sensor networks is important and challenged by the dynamic environment and its own characteristic, frequently sleeping. Although there has existed various approaches, they still suffer from message control redundancy and energy balancing issues. In this paper, we consider the problem of lightweight code distribution and energy balancing in wireless sensor networks by utilizing shared requests to reduce redundant control messages. Additionally, our contribution is enhanced by various solutions, such as multisegment advertisement strategy and edge-oriented strategy. Through analysis and evaluation sections, we confirm that our protocol does not only help to reduce update completion time by 1/3 compared with the Deluge protocol, but also significantly decreases redundant control messages and balances energy consumption among nodes in the updating process.

1. Introduction

Wireless sensor networks (WSNs) are becoming increasingly necessary in various fields due to the importance of sensor applications such as monitoring systems and warning systems [1]. To achieve useful practice, a WSN must have the ability of operating correctly for a long period of time. This introduces several challenges. Firstly, the dynamic environment changing requires WSNs to adapt and rise to the need for reliable reprogramming of all sensor nodes [2, 3]. To be more specific, nodes in a WSN should be frequently reconfigured with new updating code which is originated from one or more sinks [4]. However, due to nature of energy constraints of WSNs, the code updating process must be optimized. Secondly, a WSN can operate incorrectly as if some nodes die out. Therefore, it should keep energy balancing among sensor nodes to avoid those nodes running out of energy [5].

To address the above difficulties, various code distribution protocols have been proposed. Some approaches are for special sensors such as Sprinkler [6] and GARUDA [7]. In these protocols, at the first step, based on the location information of each sensor, some sensor nodes are selected to receive the update information to achieve energy optimization. Then, in the next step, these nodes spread out that information to all nodes. However, the location information is not commonly supported for sensors. Therefore, most of the available approaches are focused for normal sensors. Traditionally, reprogramming is done within the range of a sink in Mica-2 mode [8] in which a node keeps an update information and other nodes in the range of that node. However, this method seems to be impractical due to the nature of WSN spreading over a large area. Therefore, the initial version of multihop reprogramming was presented by Kulik et al. in 2002. According to this protocol, Kulik et al. took advantage of the advertisement-request-data model in an epidemic algorithm in SPIN protocol [9]. A node after receiving the full version of update information will become a sender to its neighbors. However, overlapping among the transmission range of sensors results in a control message redundancy problem at over 95%. To overcome this obstacle, the Trickle [10] protocol is proposed in 2004 by using SRM-like suppression mechanism. Later, Hui and Culler integrated Trickle into the Deluge protocol [11]. Because of the large amount of update information, therefore, start from sink, this information is divided into multiple parts. Each part of the message, then, used advertisement-request-data model to spread to all sensor nodes. It is estimated as a highly reliable, optimized, code-dissemination protocol. Although providing various methods of suppressing request messages, Deluge still generates too much unnecessary advertise messages due to
frequently sending them. Additionally, the performance analysis section later indicates that the update completion time of the whole network increases exponentially while the size of the network gets bigger. As a result, it does not satisfy the energy constraints in WSNs. Moreover, previous approaches did not consider the energy balancing problem. Nodes in the core of a network spend much more energy than nodes on the edge in the activities of sending and receiving packets. For instance, the result in the Deluge paper shows that within the interference range of a node at the edge a much larger number of messages were sent than that of a node at the core.

The above obstacles challenge not only Deluge, but also various next generations of this algorithm, such as MNP [12], FRESHET [13], SYNAPSE++ [14], and Typhoon [15]. Although each paper proposes a solution for a new problem, it still suffers from the above problems of the foundation transmission model (advertise-profile-request-data). Take the FRESHET protocol as the typical example. This protocol includes three phases: warming phase, distribution phase, and quiescent phase. The idea here is reducing energy consumption by providing a warming phase. Once an update version for sensor nodes is available, the sink will send the informing message to all nodes. Most of nodes which are far from the sink are put into sleep state. Only these nodes which are near the sender will be active in order to receive the update information. The next two phases are inherited from the Deluge protocol. Take another example, the solution in MNP protocol is trying to reduce to collision and hidden terminal by selecting a suitable sleeping schedule among sensor nodes. A node is in the sleep mode as if it receives an unnecessary message. These protocols only focus on sleep schedule of nodes but do not focus on the methods to reduce the redundant messages.

To cope with the above challenges, then, we introduce a protocol named Dsare in which we provide various mechanisms to minimize redundant control messages and quickly distribute code by proposing a new code dissemination model. Afterwards, the contribution is enhanced by providing the edge-oriented strategy to cope with the energy balancing problem. Additionally, an analysis for update completion time and a variety of simulation results concretely confirms the advantages of our solution over the Deluge protocol. Accordingly, the time required to finish the update process is reduced by 1/3, compared with that of Deluge, and there is a dramatic reduction in the proportion of control message redundancy, at around 80% in Deluge and lower than 25% in the Dsare protocol.

The rest of this paper is categorized into three sections. Section 2 firstly describes the basic model of our protocol. Then, some details of our proposed transmission model and enhancements are presented. The next section analyzes the update completion time of our protocol, Dsare. The last section covers the diversity of results of our protocol in comparison with the Deluge protocol and the specific characteristics of Dsare.

2. Code Dissemination Protocol

2.1. Transmission Model. Dsare is an epidemic protocol and follows the advertise-request-data model [11]. In this protocol, utilizing the broadcast medium, all the packets are shared, and especially the shared request messages are seriously considered. First of all, the basic form of the Dsare protocol is illustrated as in Figure 1. The model follows the advertise-request-data model. However, the shared request messages help to reduce the large number of advertise messages. In detail, initially, a sink with a new version image that acts as the first sender broadcasts the new image information to all sensor nodes within its range. After one-hop-count neighbors of the sink receive the advertise message, they decide to send a request message back to the originator. Then, the sender broadcasts a new version image to all of its neighbors. At the same time, two-hop-count neighbors of the sender overhear that request message and estimate an interval before sending a request back to the one-hop-count neighbor to achieve a new version image. After obtaining new data, a one-hop-count neighbor becomes a new sender. And the process is repeated until all nodes are updated completely.

At this intuitive version of the Dsare protocol, request messages are shared at the maximum to minimize unnecessary control messages. By overhearing request messages, the two-hop-count neighbors know one-hop-count nodes are going to obtain new data messages. Therefore, it sends a request back to the one-hop-count nodes without receiving an advertise message from that node after a predetermined interval of time. As a result, advertisement messages are mostly saved.

Clearly, the basic form of Dsare is relatively simple. And some obstacles do challenge it. One of them is request missing, which is the result of the difference between the rate
of distributed request messages and that of data messages
and the loss link. Another is the requirement of a suitable
method to calculate the waiting interval at the two-hop-
count neighbor. Therefore, in the next section, various critical
issues in WSNs are considered seriously to enhance overall
performance of the protocol. Firstly, it minimizes advertise
and request messages. Secondly, it is about energy balancing.
Thirdly, link fails are also considered.

2.2. Multisegment Advertisement. If only the above basic
model is applied, a new problem is appearing that a node
sends request message to obtain the data while all of its
neighbors do not have the new update information due to the
difference between the rate of disseminating request messages
and rate of distributing data. For instance, once there is an
error while transmitting data, a node is unable to obtain
data. But, at the same time, one of its neighbors start to send
request message to this node while it is not ready to send
data. Therefore, in order to eliminate this issue, Dsare applies
a strategy to control the schedule of advertise and request
messages. To be more specific, all nodes that are \( k \) * BOUND-
hop-count far from the sink have to broadcast an advertise
message (with \( k = 0, 1, 2, \ldots \) and BOUND is a predetermined
number). Nodes that are \((k \times \text{BOUND} + 1)\) hop-count far from
the sink should only send a request message after receiving
an advertise message instead of receiving the previous node’s
request message and calculating the waiting time. Therefore,
the mentioned difference between two rates is no longer
relied to the next BOUND-hop-count nodes.

2.3. Edge-Oriented Strategy. A new problem is that some
nodes have to spend much of the energy for requesting and
data sending while other nodes do not. Then, these nodes
deplete quickly. Therefore, in response to energy balancing,
Dsare also applies an edge-oriented strategy, which is a
strategy that chooses a sender among the various senders that
broadcast new image information to that node. Accordingly,
each node in the network estimates the number of neighbors
by overhearing request and advertise messages sent by its
neighbors. Then, this information is passed to its neighbors
through either advertise or request messages. A node calculates
the density of the local area based on that data. Finally,
the node then selects the best sender with the lowest density
value.

2.4. The Protocol. A node running the Dsare protocol transits
among three states: MAINTAIN, REQ, and TRANS. All
nodes follow a set of rules in response to incoming events.
An update image is segmented into multiple pages, as in the
Deluge paper.

2.4.1. Maintenance Phase. A node is in the MAINTAIN state
when it responds, informing of the availability of a new
version, or in the process of receiving a new image. For
the former purpose, a node must be the sink or satisfy the
condition of a multisegment advertisement strategy, or not
send any request message, right after it receives a page of the
new version image. Then, after an interval time \( T_{\text{adv}} \) expired,
the node calls an advertise message function

\[
T_{\text{adv}} = \varphi + \tau_r,
\]

where \( \varphi \) is denoted as a predetermined interval and \( \tau_r \)
is denoted as a random time interval to avoid collision. After
sending advertise message, the state of the node is transited
into REQ.

For the latter one, if a node is in the process of receiving a
data message, it is transited to a MAINTAIN state.

2.4.2. Request Phase. A node is in REQ state when it needs to
update a new version message. At that moment, that node has
just received a new advertise or request message with version \( \theta' > \theta \), which is the current version of this node, and the
incoming page \( y' > y \), which is the highest updated page
of this node. Whenever a node updates a page, it will send
a request to update the next page of the image.

Handling the request function is the most complicated
function in this program. There are two options. One is after
obtaining a request message; if this node has a new version
image, it sends data back and transits the state to TRANS.
In other case, this node sends the request message back
as if the number of hop counts from this node to the sink is
different from the \((\text{BOUND} \times k + 1)\) hop count.
A request is sent back after the interval:

\[
T_{\text{req}} = \text{CONST}_{\text{REQTIME}} + \tau_r.
\]

However, the interval for the request to be sent again if
the first request failed must be different to take advantage of
multiple neighbors in WSNs:

\[
T_{\text{req}} = \omega + \tau_r.
\]

2.4.3. Transition Phase. This state is responsible for broad-
casting new version image packets from a sender to its
neighbors. Whenever a node receives a data packet, if this
node is not updated yet, it should receive all data messages,
regardless of its current state.

3. Performance Analysis

To have an objective point of view about the performance
of our protocol, this section focuses on analyzing the update
completion time of Deluge and Dsare by using mathematical
calculation.

Firstly, we consider the Deluge algorithm by taking into
account node B and node C. B and C are \( i \)-hop-count and
\((i + 1)\)-hop-count nodes far from the source, respectively. C
is B’s neighbor. Figure 2 illustrates the activities of sending
and receiving messages of two nodes from the time a node
receives an advertise message until that node receives the new
version of the message. \( T_i \) and \( T_{i+1} \) are denoted as the intervals
before receiving the new version profile messages at nodes B
and C, respectively. \( T_i \) is the shortest interval in which it is set
whenever a node has a different version with at least one of its neighbors. According to the Deluge algorithm,
\[ T_i = T_{i+1}, \]
\[ T_{i+1} = T_{i+2}. \]

Assume that all messages are 100% successfully received. \( \Delta t \) represents the interval of successfully receiving data messages between 2 neighbors. Then,
\[ \Delta t \approx T_{i+1} = T_{i+2}. \]

According to (5) above, if \( T_i \) is considered a constant value, \( \Delta t \) depends largely on the distance between this node and the sink. A node (e.g., node C) is updated for a longer time compared to its sender (node B) right before the value of \( i \) gets higher. Then, the completion time of the whole network increases. In other words, it suffers scalability and energy-constrained problems.

In the remainder of this section, we discuss the update completion time of a node running the Dsare protocol.

Figure 2(b) describes the schedule of sending and receiving messages in a Dsare node. Message 2 can be an advertise message or an overhear message.

If we consider the normal case, as mentioned above, in which there is no request miss, then nodes B and C are quickly updated:
\[ \Delta t \approx T_{\text{req}}. \]

This also means that the update completion time for the whole network is much smaller in the case of Dsare when comparing (5) and (6). However, we consider more deeply, where node B requests failed \( n \) times before being updated,
\[ \Delta t \approx T_{\text{req}} + n \cdot T_\omega. \]

At the node \( D \ h \)-hop count far from node B,
\[ \Delta t \approx T_{\text{req}} + \sum n \cdot T_\omega. \]

From (2), (3), and (8), \( \Delta t \) depends largely on the number of rerequests. That is the reason why multisegment advertisement strategies, along with BOUND values, are taken into account to limit the upper bound of \( \Delta t \). According to the multisegment advertisement strategy, \( \Delta t \) is reset to 0 each time the condition of the strategy is satisfied.

However, one of the specific weak points of this protocol, or the weak point of the model advertise-request-data in general, is that there is one advertise message along with the direction from the sender to the receiver. Although a node can take advantage of multiple neighbors to enhance this weak point, there could still be losses in the update process, in the worst case.

4. Performance Evaluation

This section focuses on measuring the overall performance of our protocol by simulating it on the ContikiOS and
Cooja simulator and comparing the results with those of the Deluge protocol. To achieve this objective, we consider various metrics and use them as the main characteristics to evaluate our protocol. Some of them were already used in the Deluge paper, such as completion reliability, completion time, and energy consumption. Some new metrics are also exploited:

(i) Number of control messages: in the Deluge case, it includes advertise, profile, and request messages, whereas, in the Dsare case, it includes advertise and request messages.

(ii) Number of congestion points $N_{co}$: it is the number of connection between 2 nodes in which collision occurs.

(iii) Proportion of redundant control messages: It is equal to the division of the number of redundant message $N_{dup}$ by the total number of generated message at a node, where

$$N_{dup} = \sum \text{all messages} - \sum N_{ne} - \sum N_{co} \quad (9)$$

where $N_{ne}$ is the necessary message to transmit a page of data message. Consider transmission from node A to node B in the Deluge protocol:

$$N_{ne} = \text{data message} + 2 \text{ advertise message} + 1 \text{ request message} + 1 \text{ profile message}. \quad (10)$$

Now consider it in the Dsare protocol:

$$\sum N_{ne} = \sum \text{data message} + \sum \text{advertise message} + \sum \text{request message per node}. \quad (11)$$

(iv) Energy balancing: the balance among the total energy consumption of each node.

4.1. Simulation Environment

4.1.1. ContikiOS and Cooja. We rely on the Contiki operating system (OS) [16] to evaluate the overall performance of our protocol. Each virtual sensor is installed in this OS. C language is used as the primary language to build this OS. Applications are written by this language and are called through an interface of the Cooja simulator. This simulator is a discrete-event network simulator and is compiled directly from the ContikiOS.

4.1.2. Parameter Configuration. In each simulation, nodes are located in a square network $N \times N$ ($N \in [2, 20]$). A sink is located at the corner of the network. At the initial stage, all nodes are powered up and contain image version 0, except the sink node that has version 1. Each image version contains just one page [11]. Each page includes 8 packets. Each packet is 64 bytes. Thus, the total size of the image is 500 bytes. After a random interval time $T_r$, the sink node broadcasts the new version information to start the update process. Each node is configured with $\text{BOUND} = 4$, $\text{CONST}_\text{REQTIME} = 4$, $T_r = 0.5$, $\lambda = 4$, $\omega = 8$.

4.2. Simulation Results. This section briefly describes the various results of the Dsare protocol and makes some comparisons with the Deluge protocol. These results are categorized into two groups. One is proving the efficiency of our protocol compared with that of the Deluge protocol. The other illustrates specific characteristics of Dsare.

For the first group, we investigate two aspects. One is the interval time to finish the updating process in all sensor nodes. Another is the cost for the updating process.

4.2.1. Time to Complete. Figure 3 illustrates the completion time of the Deluge and the Dsare protocol in a square network $N \times N$ ($N \in [2, 20]$). Node density is kept constant at 10 meters apart. According to this line graph, the figures for both protocols gradually increase. The figures for Dsare are around 2/3 of those for Deluge. At the network scale $20 \times 20$, the completion time for the Deluge protocol is approximately 290 seconds, whereas for Dsare it is just around 200 seconds.

Next, we compare the number of updated nodes between the above protocols in a specific topology at different periods. In this case, we choose the topology $20 \times 20$. Figure 4 shows that Dsare has a trend to spread from nodes at the border to the center node.

4.2.2. Cost of Updating Process. From the above comparison, it is easy to recognize that the Dsare protocol helps sensor nodes update more quickly than the Deluge protocol does.

However, the efficiency of the Dsare protocol also depends on the cost of paying for the updating process. This cost is considered the compound of the number of control messages, the number of redundant messages, and the number of collisions that occur in the network.

The first considered attribute is the number of control messages. The method of calculating this number is
mentioned in the previous section, to compare 2 algorithms in square topologies with \( \text{BOUND} = 2 \). According to Figure 5, the figure for Deluge dramatically increases whereas the figure for the Dsare protocol gradually rises. When the diameter of the network is small, the difference between the two figures is around 8 times. However, when the diameter enlarges, the number of control message for Deluge reaches 10070 and is more than 13 times that of Dsare.

As mentioned, the proportion of collisions is also an important characteristic to contribute to the efficiency of a protocol. Motivated from the previous results, we took various implementations and obtained some achievements, as shown in Figure 7. Obviously, the number of collisions that occurs in a network increases with the rise in the size of the network. However, the figure for Dsare is around 60%, compared with that of Deluge.

After comparing the number of control messages between the two protocols, we look deeper into the proportion of control message redundancy and data message collision. Figure 6 shows that while the proportion of Deluge redundant control messages is relatively stable at around 80%, the figure...
Figure 7: Comparing the number of collision points between Dsare and Deluge protocol with different network parameters.

Figure 8: Comparison number of request misses between Dsare protocol with various values of BOUND and Deluge protocol.

Figure 9: Comparison of energy balancing between two protocols in terms of radio transmit time in topology $10 \times 10$.

4.2.3. Energy Balancing. To evaluate two protocols in terms of energy balancing, various simulators take place in a square topology $10 \times 10$. Energy balancing is reflected in the balance among the total energy consumption of each node. The total energy consumption depends on four main aspects: full CPU energy consumption, reduced power CPU consumption, radio receive, and transmit power consumption. 

However, the three former gradually increase due to the assumption that all nodes in a network stay awake until finishing the update process. Then, the difference in terms of power consumption between two protocols depends on the difference in transmitted power consumption between those protocols. This parameter depends on the total radio transmit time; thus, we calculate the radio transmit time of 100 nodes when sequentially applying two code distribution protocols. The result is illustrated in Figure 9. Obviously, nodes using the Dsare protocol show their energy balance as better than that of Deluge.

5. Conclusion

In this paper, we presented various improvements to address control message redundancy and energy balancing problems in reprogramming wireless sensor networks. By utilizing shared requests, applying multisegment advertisement strategy and edge-oriented strategy, and, later, through analyzing various simulations, we experienced that our protocol update completion time equals $2/3$ that of the Deluge protocol, with the number of required messages reduced by more than 10 times. The energy for the updating process is balanced among nodes. Furthermore, we addressed the emerging problem as if our protocol was applied. It is the different distributed rate between the request message and the data message.
Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the C-ITRC (Convergence Information Technology Research Center) (IITP-2015-H8601-15-1001) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] F. Wang and J. Liu, “Networked wireless sensor data collection: issues, challenges, and approaches,” IEEE Communications Surveys and Tutorials, vol. 13, no. 4, pp. 673–687, 2011.
[2] Q. Wang, Y. Zhu, and L. Cheng, “Reprogramming wireless sensor networks: challenges and approaches,” IEEE Network, vol. 20, no. 3, pp. 48–55, 2006.
[3] W.-K. Chen, Linear Networks and Systems, Wadsworth, Belmont, Calif, USA, 1993.
[4] P. E. Lanigan, R. Gandhi, and P. Narasimhan, Disseminating Code Updates in Sensor Networks: Survey of Protocols and Security Issues, Carnegie Mellon University, 2005.
[5] H. Zhang and H. Shen, “Balancing energy consumption to maximize network lifetime in data-gathering sensor networks,” IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 10, pp. 1526–1539, 2009.
[6] V. Naik, A. Arora, P. Sinha, and H. Zhang, “Sprinkler: a reliable and energy efficient data dissemination service for wireless embedded devices,” in Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS ’05), December 2005.
[7] S.-J. Park, R. Sivakumar, I. F. Akyildiz, and R. Veidantham, “GARUDA: achieving effective reliability for downstream communication in wireless sensor networks,” IEEE Transactions on Mobile Computing, vol. 7, no. 2, pp. 214–230, 2008.
[8] J. L. Hill and D. E. Culler, “Mica: a wireless platform for deeply embedded networks,” IEEE Micro, vol. 22, no. 6, pp. 12–24, 2002.
[9] J. Kulik, W. Heinzelman, and H. Balakrishnan, “Negotiation-based protocols for disseminating information in wireless sensor networks,” Wireless Networks, vol. 8, no. 2-3, pp. 169–185, 2002.
[10] P. Levis, N. Patel, S. Shenker, and D. Culler, “Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” Tech. Rep., University of California, Berkeley, Calif, USA, 2004.
[11] J. W. Hui and D. Culler, “The dynamic behavior of a data dissemination protocol for network programming at scale,” in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys ’04), pp. 81–94, ACM, November 2004.
[12] S. S. Kulkarni and L. Wang, “MNP: multihop network reprogramming service for sensor networks,” in Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS ’05), pp. 7–16, June 2005.
[13] M. D. Krasniewski, R. K. Panta, S. Bagchi, C.-L. Yang, and W. J. Chappell, “Energy-efficient on-demand reprogramming of large-scale sensor networks,” ACM Transactions on Sensor Networks, vol. 4, no. 1, article 2, pp. 1–38, 2008.
[14] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, “SYNAPSE++: code dissemination in wireless sensor networks using fountain codes,” IEEE Transactions on Mobile Computing, vol. 9, no. 12, pp. 1749–1765, 2010.
[15] C.-J. M. Liang, R. Musáloiu-E, and A. Terzis, “Typhoon: a reliable data dissemination protocol for wireless sensor networks,” in Wireless Sensor Networks: 5th European Conference, EWSN 2008, Bologna, Italy, January 30-February 1, 2008. Proceedings, vol. 4913 of Lecture Notes in Computer Science, pp. 268–285, Springer, Berlin, Germany, 2008.
[16] Contiki OS, Stockholm, Sweden, April 2013, http://www.contiki-os.org/.
