Non-Invasive Rapid and Efficient Firmware Update for Wireless Sensor Networks

Hui Ung Park
Korea University of Science
and Technology, Rep. of Korea
qyan@etri.re.kr

Jongsoo Jeong
Electronics and
Telecommunications Research
Institute, Rep. of Korea
jsjeong@etri.re.kr

Pyeongsoo Mah
Electronics and
Telecommunications Research
Institute, Rep. of Korea
Korea University of Science
and Technology, Rep. of Korea
pmah@etri.re.kr

Ubicomp '14, Sep 13-17 2014, Seattle, WA, USA
ACM 978-1-4503-3047-3/14/09.
http://dx.doi.org/10.1145/2638728.2638782

Abstract
To maintain software of sensor nodes in wireless sensor networks efficiently, it is necessary to minimize the size of transferred data in firmware update. We propose a non-invasive rapid and efficient incremental firmware update algorithm called MoRE. In MoRE algorithm, the host transfers only delta, which is the information of different parts between old and new firmware image, to reduce the size of transferred data. The sensor node makes new binary image from its current image and the transferred messages. The MoRE shows comparable performance to previous works without invasive methods. Unlike the previous works, MoRE does not require extra memory for metadata in sensor nodes and does not need to use relocatable code.

Introduction
The network connection of all things through the Internet, called Internet of Things (IoT), became a trend in wireless sensor networks (WSNs). To deploy the Internet-based WSNs, a number of IETF protocols are being standardized [5]. As many protocols are applied, the sensor node software become heavier and more complicated. Consequently, the maintenance of sensor nodes' software becomes difficult. Since most wireless sensor nodes have cheap communication modules with low bandwidth, they are not suitable for massive wireless

transmissions. For these reasons, it is required to minimize the size of transferred data.

Related work
Previous researches have attempted to reduce the size of transferred data by incremental reprogramming [1, 2, 4]. The incremental reprogramming uses delta which is the differences between two version of a firmware image. In general, users update sensor nodes' firmware to apply bug fixes or changed mission. The modified part of the firmware image by the update usually occurs locally. If the sensor node reconstructs the new image by using the delta and the old image, the size of transferred data can be reduced significantly.

The host transfers two types of messages in incremental reprogramming shown in Figure 1. *Copy message* instructs the sensor node to copy a raw binary data from current binary image of the sensor node to target address. *Down message* contains delta with target address.

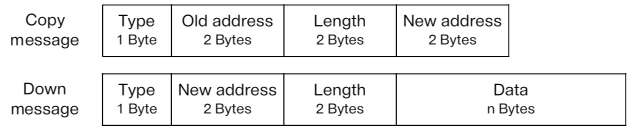

Figure 1: The structure of *copy message* and *down message*.

Prior researches tried to reduce the size of the delta. RMTD [2] finds common segments (CSs) of two binary codes in byte-level and calculates optimal combination of CSs. RMTD algorithm finds CSs not only from previous binary code and new binary code, but also from new binary code and partially transferred new binary codes. So, sensor node can make whole new binary code from partially transferred new binary code. However, RMTD algorithm has $O(n^2)$ space complexity and it results in memory limitation problem due to its full scan manner. Hermes [4] manages indirection table to mitigate the effect of function shift and global variable shift. These tables should be stored into sensor node's memory. R3 [1] uses the firmware that is relocatable code to maximize the similarity between two versions of codes. R3 also has an optimal byte-level delta generation algorithm called R3diff. Users who want to use Hermes or R3 must change their binary code into relocatable code. Since they are invasive, they could not be adopted widely in commercial systems.

The size of input binary data influences the delta calculation sequence significantly. The time complexity of both RMTD and R3diff is $O(n^3)$ and the space complexity of RMTD and R3diff is $O(n^2)$ and $O(n)$ respectively, where n is the size of binary code. We assumed that calculating delta based on only the modified part of the binary will reduce the calculation time.

In this paper, we present a modularity-based rapid and efficient delta generating algorithm, called MoRE that does not require extra memory for metadata in sensor node and invasive modification on the code. We will show the feasibility of our algorithm in terms of implementation and efficiency perspectives.

Design

A binary firmware is made of many object files. MAP file is a text file format that contains relative offset and length of each object in binary image. A set of binary fragments that are called as modules can be extracted from an entire firmware image by using this file. MoRE consists of three sub algorithms as shown in Figure 2.

Module extraction
At the first step, the informations for identifying each modules are extracted from previous and new version of

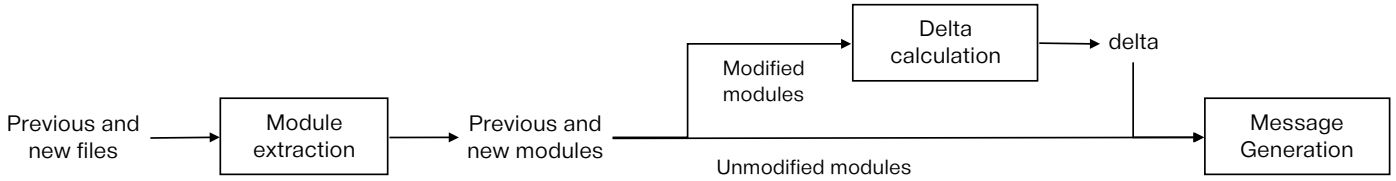

Figure 2: Workflow of MoRE algorithm.

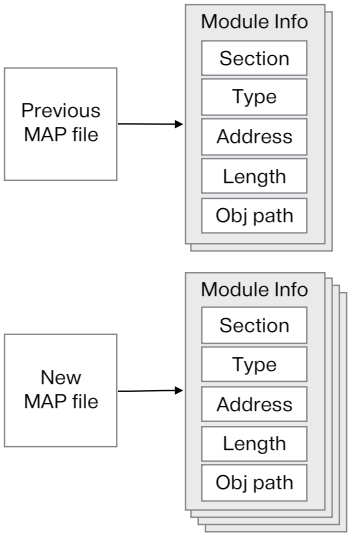

Figure 3: We could extract module metadata such as *section*, *type*, *address*, *length* and *object file path* from the firmware's MAP file.

firmware's map file as shown in Figure 3. The informations specify the particular positions in the binary firmwares to each module. After the module information is extracted, the previous and new version of firmwares are spliced into sets of module binaries based on the given module information.

Delta calculation
Once modules are extracted, the delta is calculated from each module. Modified RMTD algorithm and modified R3diff algorithm are used for the delta calculation. Only some part of RMTD algorithm that finds CSs between new image and partially constructed image is used to calculate delta from newly added modules. The size of transferred data can be reduced by the algorithm when the update contains newly added object files or libraries. Modified R3diff is optimized to calculate optimal delta for modified modules that exist in the previous binary firmware without relocatable code.

Message generation
Copy messages and *down messages* are created based on the delta. Since a plurality of delta calculation algorithms are used, the structure of both types of messages is slightly modified to identify each algorithm.

Experimental results

NanoQplus OS [3][1] and Mango-Etoi board [2] have been designated as the target platform. To evaluate the performance of MoRE, following software change scenarios on NanoQplus were set. We compared the delta calculating time and the number of bytes to be transferred.

- Case 1 (Tiny update) : The Blink application that blinks LED every 1 second is changed to blink LED every 2 seconds.
- Case 2 (Minor update) : The Blink application is changed to a button interrupt test application.
- Case 3 (Major update) : The Blink application is changed to 6LoWPAN router.
- Case 4 (Minor update in large file) : The length of 6LoWPAN router's kernel task queue is changed from 8 to 16.

[1] https://bitbucket.org/nanoqplus/nanoqplus
[2] http://www.mangoboard.com

	RMTD	R3diff	MoRE
case 1	30686	641	234
case 2	24904	62	38
case 3	100246	71	767
case 4	Fault	2989	32

Table 1: Delta calculation time (Millisecond)

	Raw	RMTD	R3diff	MoRE
case 1	18484	36	32	39
case 2	17592	4218	3931	4179
case 3	58868	48412	47591	47772
case 4	58736	Fault	408	508

Table 2: Transfer size (Byte)

The results show that MoRE took less than 1 second in delta calculation time as shown in Table 1. In case 4, while RMTD algorithm can not calculate the delta and R3diff takes almost 3 seconds to calculate it, MoRE just takes 32 milliseconds. Notice that the size of the messages generated by MoRE is almost same as the results of other algorithms as shown in Table 2.

Without any invasive methods, MoRE shows such a comparable performance. MoRE does not have to either transmit additional data such as bitmap files or tables to sensor nodes or store additional data in the memory of the sensor node that are additional burdens of invasive methods. Moreover, there is no need to replace the structure of a binary firmware to the relocatable code. It is therefore easy to apply to real commercial systems.

Conclusion

In this paper, we presented a non-invasive incremental reprogramming algorithm for firmware update, called MoRE. It uses a non-invasive module extraction method to resolve OS or toolchain dependency. Without any invasive methods, it shows comparable performance to previous works. We plan to implement MoRE and integrate with NanoQplus's firmware update tool.

Acknowledgment

This work was supported by the IT R&D program of MSIP/KEIT, Rep. of Korea (10047067, Development of SMP RTOS technology for high performance and realtime multicore embedded systems)

References

[1] Dong, W., Mo, B., Huang, C., Liu, Y., and Chen, C. R3: Optimizing relocatable code for efficient reprogramming in networked embedded systems. In *INFOCOM, 2013 Proceedings IEEE* (April 2013), 315–319.

[2] Hu, J., Xue, C., He, Y., and Sha, E.-M. Reprogramming with minimal transferred data on wireless sensor network. In *Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on* (Oct 2009), 160–167.

[3] Jeong, J., Kim, J., and Mah, P. Design and implementation of low power wireless ipv6 routing for nanoqplus. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on* (Feb 2011), 966–971.

[4] Panta, R., and Bagchi, S. Hermes: Fast and energy efficient incremental code updates for wireless sensor networks. In *INFOCOM 2009, IEEE* (April 2009), 639–647.

[5] Watteyne, T., Molinaro, A., Richichi, M., and Dohler, M. From manet to ietf roll standardization: A paradigm shift in wsn routing protocols. *Communications Surveys Tutorials, IEEE 13*, 4 (Fourth 2011), 688–707.