Applied Mathematics & Information Sciences 5 (2) (2011),89S-96S
– An International Journal

EEORP: Energy-Efficient Online Reprogramming Protocol for Wireless Sensor Networks

Mande Xie
College of Computer Science & Information Engineering, Zhejiang Gongshang University, Hangzhou, Zhejiang, People's Republic China, 310018

Email Address :xiemd@zjgsu.edu.cn

Received January 22, 2010; Revised October 21, 2010

An energy-efficient online reprogramming protocol called *EEORP* is proposed. Based on Deluge framework, *EEORP* introduces two methods to reduce the consumption of energy during the dissemination of an image. Firstly *EEORP* separates the reprogramming protocol as an alone program image and implements the application as another program image equipped with the ability to listen to new code updates. During program image dissemination, only the application program image needs to be disseminated. In addition *EEORP* improves the dynamic adjustment algorithm of transmit rate of advertisement to reserve the energy. Our algorithm is based on Deluge. Hence it can easily be integrated into Deluge and has good prospects.

Keywords: Reprogramming, Wireless Sensor Networks, Programming Image.

1 Introduction

Due to the potential to provide fine-grained sensing and actuation at a reasonable cost, wireless sensor networks (WSNs) are considered ideal candidates for a wide range of applications such as industrial monitoring and military operations. Program image updates have become very necessary in WSNs because they may be required for bug fixing or to provide new functionalities after WSNs have been deployed. However, if WSNs are large scale or deployed in a harsh environment, it is impossible to reprogram manually all nodes. Online network reprogramming can remotely reprogram all nodes via wireless communication. Hence it becomes a promising technique.

For large WSNs, where the sink cannot reach every node through broadcasting, updates can only be transmitted hop-by-hop within the WSN and this consumes significant energy. Recent studies have shown that sending a single bit of data consumes about the same energy as executing 1000 instructions [1]. However, sensor nodes are usually powered by a battery and have limited energy. As a result it is essential to conserve energy in a WSN during the dissemination of code and data, especially when the update happens frequently. At present many researchers are paying much attention to the reprogramming algorithms which minimize energy consumption. Their main

approaches are to optimize the transmitted object.

In this paper an energy-efficient online reprogramming protocol called *EEORP* is proposed. Based on Deluge framework, which has become a de facto standard of the reprogramming protocol because of the widespread incidence of Deluge, *EEORP* split the original program image into two program images and only one program image among them needs to be disseminated. As a result the transmitting energy is reduced. In addition *EEORP* employs the dynamic adjustment mechanism of transmit rate of advertisement to reserve the energy. The rest of this paper is structured as follows. Section 2 introduces related work. Section 3 presents the changes from Deluge. In Section 4 an energy-reserved strategy by splitting program image in *EEORP* is described. Dynamic adjustment mechanism of transmit rate of advertisement in *EEORP* is proposed in Section 5. At last, the brief summary is drawn in Section 6.

2 Related Works

The early network reprogramming protocol includes XNP [2] and MOAP [3]. XNP only operates over a single hop and does not provide incremental updates of the code image. The Multi-hop Over the Air Programming (MOAP) protocol extends it to operate over multiple hops. MOAP introduces several concepts which are used by later protocols. However, it does not leverage the pipelining effect with segments of the code image. Another three protocols that are substantially more sophisticated than the rest are Deluge [4], MNP [5] and Freshet [6]. All use the three way handshake for locally propagating the code. Deluge is the earliest and lays down some design principles used by the other two. It builds on top of Trickle [7] which is a protocol for a node to determine when to propagate code in an one-hop case. The design goal of MNP is to choose a local source of the code which can satisfy the maximum number of nodes. They save energy by turning off the radio of nonsender nodes. Freshet aggressively optimizes the energy consumption for reprogramming.

The literature [8-10] proposes several methods to optimize the transmitted object. The Stream [8] uses the facility of having multiple code images on a node and switches between them. Stream preinstalls the reprogramming protocol as one image and the application program equipped with the ability to listen to new code updates as the second image. Rajesh et al. propose two incremental reprogramming algorithms Zephyr [9] and Hermes [10] to minimize energy consumption. Zephyr transfers the delta between the old and the new software and reduces the delta size by using application-level modifications to mitigate the effects of function shifts. At the same time it compares the binary images at the byte-level with a novel method to create small delta. Based on Zephyr, Hermes reduces the delta by mitigating the effects of function and global variable shifts caused by the software modifications. Li [11] studies the code distribution problem in multi-

application wireless sensor networks (MA-WSNs) and proposes MCP, a Multicast based Code redistribution Protocol for achieving energy efficiency. In addition Li [12-13] proposes a novel update-conscious compilation (UCC) technique to achieve energy efficiency. By integrating the compilation decisions in generating the old binary, an update-conscious compiler strives to match the old decisions in order to reduce the amount of data transmitted to remote sensors and thereby consumes less energy. According to mobile sensor network, Pradip et al. [14] propose an energy-efficient, multihop reprogramming protocol to consider the prohibitive factor of uncertainty about a node's location due to its continuous movement.

3 Changes from Deluge

Deluge is a classic increment code dissemination algorithm which transmits the difference between the old and new program images instead of the whole program image. Deluge firstly divides the code image into a series of fixed-size pages and each page is further split into a series of same-size packets, one of which is the basic transmission unit. In Deluge the pages are propagated in sequential order, that is to say, the next page is prohibited from being requested before the previous page is completely received. However, the packets in a page can be requested out of order. After a requesting node receives all packets in a page, it can immediately advertise the availability of the newly received page and may transmit the corresponding packets upon request. Deluge employs a three-stages (advertise-request-updates) process to propagate the code image.

In Deluge each node periodically advertises the version of its code images and the number of pages it has for that version. For energy efficiency the advertisement rate is dynamically adjusted: if a node discovers its own advertisement is different from those received from another, it increases its advertisement rate. Otherwise, it decreases the rate. By this dynamical adjustment strategy Deluge can achieve rapid propagation during dissemination of the program image, but consumes little energy in steady state. To reduce redundant advertisement packets, a node suppresses its own advertisement packet if it finds the number of advertisement packets which contain the same information is more than a predefined threshold.

However, the dynamic adjustment strategy of transmit rate, which is employed in Deluge, is too coarse. At the same time Deluge integrates the reprogramming protocol into the application and transmits the whole image of the application program. In fact the reprogramming protocol component need almost not be updated. In this paper we elaborate the dynamic adjustment strategy of transmit rate and separate the reprogramming protocol as an alone program image in order to reduce the energy consumption during code dissemination. In the following sections we introduce them in detail.

4 Energy Reserved Strategy by Splitting Program Image

Unlike Deluge *EEORP* separates an alone reprogramming component from the whole program image, that is to say, *EEORP* implements the reprogramming protocol as an alone program image, denoted IMAGE_P, and implements the application as another program image, denoted IMAGE_A. Besides the normal application function, IMAGE_A is also equipped with the ability to listen to new code updates. Because the IMAGE_P is simple and invariable, in general IMAGE_P need not be updated. IMAGE_A may need to be updated if the application function is changed or some bugs are found. However, in general the difference between the modified program image and the old one is small. *EEORP* transmits the difference instead of the whole IMAGE_A.

Figure 1 shows the work flow of the reprogramming protocol. The source node firstly generates the delta script, which is composed of INSERT commands and COPY commands, by a byte level comparison tool [15] between the modified and old versions. Then the delta script is disseminated by the WSNs. Upon receiving the delta script a sensor node stores it and rebuilds the new application with the old application and delta script (as shown in Figure 4.1 and Figure 4.2). In addition any sensor node divides the external flash into several slots and the Figure 4.2 shows the layout of external flash in the nodes, where $IMAGE\_P$ indicates the reprogramming component, $IMAGE\_A\_O$ indicates the old application version, $IMAGE\_D$ indicates the delta script and $IMAGE\_A\_N$ indicates the new application version. After a node has rebuilt the new application version and saved it, it loads the new application to program memory to run by bootloader.

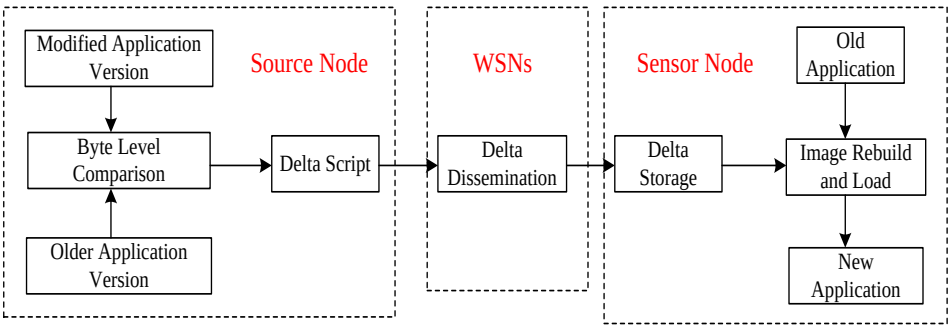

Figure 4.1: The work flow of the reprogramming protocol

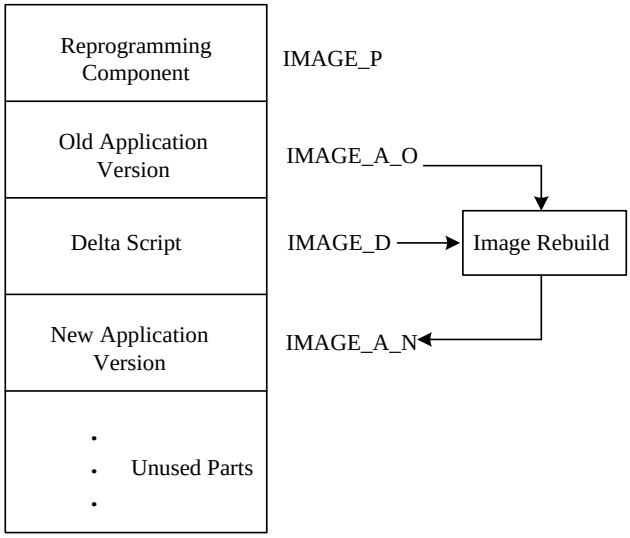

Figure 4.2: The layout of external flash in a sensor node

5 Improved Dynamic Adjustment Mechanism of Transmit Rate of Advertisement

5.1 Overview of dynamic adjustment mechanism in Deluge

Deluge uses Trickle to control the transmission of potentially redundant messages. Trickle divides time into a series of rounds and nodes to choose whether or not to broadcast an advertisement in each round. The duration of round i is specified by $\tau_{m,i}$ and is bounded by τ_l and τ_h. In each round a node maintains a random value r_i in the range $[\frac{\tau_{m,i}}{2}, \tau_{m,i}]$. During round i with start time $t_i = t_{i-1} + \tau_{m,i-1}$ broadcasts an advertisement with summary ϕ at time $t_i + r_i$ only if fewer than k advertisements with summary $\phi' = \phi$ have been received since time t_i. If any overheard packet indicates an inconsistency among neighboring nodes were overheard during round i, set $\tau_{m,i} = \tau_l$ and begin a new round. At the beginning of a round i, if no overheard packet indicates an inconsistency among neighbors during the previous round, set $\tau_{m,i}$ to $\min(2*\tau_{m,i-1}, \tau_h)$.

5.2 Dynamic Adjustment Method in *EEORP*

The dynamic adjustment mechanism in Deluge is too coarse and the effect is unsatisfactory. So we present an elaborate dynamic adjustment algorithm. Figure 5.1 shows the pseudocode description of dynamic adjustment algorithm presented by us and Table 5.1 shows notations used in Figure 3, where $TM_{i-1} = M^d_{i-1} + M^s_{i-1}$. As shown in Figure 5.1, the node adjusts the duration of $\tau_{m,i}$, based on the M^d_{i-1} and M^s_{i-1} heard by it

during round $\tau_{m,i-1}$. If $TM_{i-1} > 0$, the value of $\tau_{m,i}$ is determined by the value of M_{i-1}^{d} and M_{i-1}^{s}. In general, if M_{i-1}^{s} is larger than M_{i-1}^{d}, then increase $\tau_{m,i}$. However, if M_{i-1}^{d} is larger than M_{i-1}^{s}, then decrease $\tau_{m,i}$. In essence the final value of $\tau_{m,i}$ is a combination of these two effects which are normalized into a weighted sum. As is shown in the line (5), the value of $\tau_{m,i}$ can be computed by Eq.(5.1).

$$\tau_{m,i} = \left(\frac{M_{i-1}^{d}}{TM_{i-1}} - \delta \right) \tau_{m,i-1} + \left(\frac{M_{i-1}^{s}}{TM_{i-1}} + \frac{M_{i-1}^{s}}{M_{i-1}^{d}} \delta \right) \tau_{m,i-1} \tag{5.1}$$

```
DynAdjustAlg                                   /*Adjust the duration of τ_{m,i} */
Inputs:  τ_{m,i-1}, M_{i-1}^d, M_{i-1}^s, δ, iNoADV, MAX_NO_ADV
Outputs: τ_{m,i}
{
(1)  if (TM_{i-1} > 0) {
(2)     if ( M_{i-1}^d > M_Th)
(3)        τ_{m,i} = (M_{i-1}^d / TM_{i-1}) τ_{m,i-1} (1 + (M_{i-1}^d / TM_{i-1}) δ) + (M_{i-1}^s / TM_{i-1}) τ_{m,i-1} (1 + δ) ;
(4)     else
(5)        τ_{m,i} = (M_{i-1}^d / TM_{i-1} - δ) τ_{m,i-1} + (M_{i-1}^s / TM_{i-1} + (M_{i-1}^s / M_{i-1}^d) δ) τ_{m,i-1} ;
(6)  }
(7)  else{
(8)     iNoADV += 1;
(9)     if (iNoADV < MAX_NO_ADV)
(10)       τ_{m,i} = τ_{m,i-1} (1 - δ);
(11)    else{
(12)       iNoADV = 0;
(13)       Sleep for a short duration;
(14)       Set τ_{m,i} to the initial value;
(15)    }
(16) }
(17) if( τ_{m,i} > τ_h )
(18)    τ_{m,i} = τ_h ;
(19) if( τ_{m,i} < τ_l )
(20)    τ_{m,i} = τ_l ;
}
```

Figure 5.1: The pseudocode description of dynamic adjustment algorithm.

However, as is shown in lines (2-3), in order to avoid advertisement packet collisions nodes increase $\tau_{m,i}$ by Eq.(5.2).

$$\tau_{m,i} = \frac{M_{i-1}^{d}}{TM_{i-1}} \tau_{m,i-1} \left(1 + \frac{M_{i-1}^{d}}{TM_{i-1}} \delta \right) + \frac{M_{i-1}^{s}}{TM_{i-1}} \tau_{m,i-1} (1 + \delta) \tag{5.2}$$

If M_{i-1}^{d} crosses the threshold value of M_{th}, as shown in the line (7-16), if $TM_{i-1}=0$, that is to say, the sum of all advertisement messages heard by the node during $\tau_{m,i-1}$ is 0, $iNoADV$ adds 1. If $iNoADV < MAX\_NO\_ADV$, then decrease $\tau_{m,i}$ by Eq.(5.3) in order to increase the frequency of advertisement packet broadcast, otherwise, the node sleeps for a short duration and sets $\tau_{m,i}$ to the initial value.

$$\tau_{m,i} = \tau_{m,i-1}(1-\delta) \tag{5.3}$$

As is shown in the line (17-20), if the $\tau_{m,i}$ is larger than τ_h, $\tau_{m,i}$ is set to τ_h. Similarly, if the $\tau_{m,i}$ is lower than τ_l, $\tau_{m,i}$ is set to τ_l.

Table 5.1 Notations used in this paper

Notation	Definition
$\tau_{m,i}$	: The duration of round *i*
τ_h	: The upper bound of $\tau_{m,i}$
τ_l	: The low bound of $\tau_{m,i}$
M_{i-1}^{d}	: The number of different advertisements messages heard by a node during $\tau_{m,i-1}$.
M_{i-1}^{s}	: The number of similar advertisements messages heard by a node during $\tau_{m,i-1}$.
TM_{i-1}	: The sum of all advertisement messages heard by the node during $\tau_{m,i-1}$.
δ	: A small step used to increase or decrease the duration.
$iNoADV$	: The number of round with no received advertisements
MAX_NO_ADV	: A threshold number of round with no received advertisements

6 Conclusion

In this paper an energy-efficient online reprogramming protocol called *EEORP* is proposed. Based on Deluge framework, *EEORP* introduces two methods to reduce energy consumption during image dissemination. Firstly *EEORP* separates the reprogramming protocol as an alone program image and implements the application as another program image equipped with the ability to listen to new code updates. During program image dissemination, only the application program image needs to be disseminated. As a result the transmitting energy is reduced. In addition *EEORP* improves the dynamic adjustment algorithm of transmit rate of advertisement to preserve energy. This algorithm determines the final value of $\tau_{m,i}$ by the combination of M_{i-1}^{d} and M_{i-1}^{s}. If M_{i-1}^{s} is larger than M_{i-1}^{d}, then increase $\tau_{m,i}$, otherwise decrease $\tau_{m,i}$. Deluge has become a de facto standard of the reprogramming protocol because of the widespread utilisation of Deluge. Our algorithm is based on Deluge. Hence it can easily be integrated into Deluge and has good prospects.

References

[1] V. Shnayder, M. Hempstead, R. R. Chen, G. W. Allen and M. Welsh, Simulating the power consumption of large-scale sensor network applications, ACM Conference on Embedded Networked Sensor Systems (SenSys), pp188–200, 2004.

[2] C. T. Inc., Mote In-Network Programming User Reference,http://www.tinyos.net/ tinyos-

1.x/doc/Xnp.pdf (2003).

[3] T. Stathopoulos, J. Heidemann and D. Estrin, A remote code update mechanism for wireless sensor networks, Tech. Rep. (2003).

[4] 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, Baltimore, MD, USA, pp 81-94, 2004.

[5] S. S. Kulkarni, W. Limin, Mnp: Multihop network reprogramming service for sensor networks, in: the 25th IEEE International Conference on Distributed Computing Systems, Columbus, OH, United States, pp 7-16, 2005.

[6] M. D. Krasniewski, S. Bagchi, C.-L. Yang and W. J. Chappell, Energy efficient, on-demand reprogramming of large-scale sensor networks, ACM Transactions on Sensor Networks 4 (1).

[7] P. Levis, N. Patel, S. Shenker and D. Culler, Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor network, in: Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004), Grand Hyatt, San Francisco, pp 15-28, 2004.

[8] R. Panta, I. Khalil and S. Bagchi, Stream: Low overhead wireless reprogramming for sensor networks, in: IEEE Infocom (2007), Anchorage, AK, United states, pp 928-936, 2007.

[9] K. P. Rajesh, S. Bagchi and S. Midki_, Zephyr: Efficient incremental reprogramming of sensor nodes using function call indirections and difference computation, in: the USENIX Annual Technical Conference (USENIX '09), San Diego, California, United states, pp 411-424, 2009.

[10] R. Panta and S. Bagchi, Hermes: Fast and energy efficient incremental code updates for wireless sensor networks, in: IEEE Infocom 2009, Rio de Janeiro, Brazil, pp 639-647, 2009.

[11] W.J. Li, Y. Du, Y.T. Zhang et al. MCP: An energy-efficient code distribution protocol for multi-application WSNs. In Proceedings of 5th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'09), June 2009, Marina del Rey, CA, United states, pp 259-272.

[12] W.J. Li, Y.T. Zhang, J. Yang et al. UCC: Update-conscious compilation for energy efficiency in wireless sensor networks. ACM SIGPLAN Notices, 42(6), In Proceedings of the 2007 PLDI conference, June 2007, New York, United States, pp 383-393.

[13] W.J. Li, Y.T. Zhang, J. Yang et al. Towards update-conscious compilation for energy-efficient code dissemination in WSNs. Transactions on Architecture and Code Optimization, 6(4), October 2009, pp 214-228.

[14] L. Y. Pradip, D. and S. K. Das, Energy-efficient reprogramming of a swarm of mobile sensors, IEEE Transactions on Mobile Computing 9 (5) , 2010, pp 703-718.

[15] A. Tridgell, Efficient Algorithms for Sorting and Synchronization, PhD thesis, Australian National University.

Mande Xie received the MS degree in Circuit & System from Hangzhou Dianzi University in 2003, and the PhD degree in Circuit & System from Zhejiang University in 2006. He is currently an Associate Professor in the Zhejiang Gongshang University. His research interests are in the areas of Wireless Sensor Networks(WSNs), Peer to Peer(P2P) and Distributed Systems.