Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 324632, 9 pages
http://dx.doi.org/10.1155/2013/324632

Research Article

An Efficient Resource Management Protocol for Handling Small Resource in Wireless Sensor Networks

Wan-Hee Cho, Jiho Kim, and Ohyoung Song

School of Electrical and Electronics Engineering, Chung-Ang University, Seoul 156-756, Republic of Korea

Correspondence should be addressed to Ohyoung Song; song@cau.ac.kr

Received 1 February 2013; Accepted 24 April 2013

Academic Editor: Kayhan Gulez

Wireless sensor nodes with single chips may have insufficient resources for certain applications. We propose a resource management protocol for applications with constrained resources to improve effectiveness by borrowing resources from a resource management server.

1. Introduction

Wireless sensor nodes are often used in harsh environments such as sewers, bridges, and the outer walls of buildings. Most are operated by small batteries, which mean that the power consumption and size of devices should be reduced. These requirements can be met through the use of an integrated circuit that reduces the usage of external components. The highly integrated single-chip approach is a preferred solution for semiconductor manufacturers, since the total system cost is a key factor for industrial and home wireless applications [1]. Cost considerations are driving implementations towards single-chip solutions with a minimum number of external components [2]. The single-chip approach is also advantageous for reducing device size through the use of well-designed integrated circuits to reduce the number of external components. The single-chip approach is a mainstream method used for developing wireless sensor network-compliant devices. However, the limitations of the device size and the power consumption still lead to various constraints, such as a slower microprocessor unit (MPU) and smaller memory size. The most recent single-chip designs, especially for wireless sensor nodes, provide about 4~16 Kbytes of SRAM and an 8-bit microprocessor capable of 16 million instructions per second (MIPS) [3, 4]. The available memory becomes much smaller after using up the memory for a ZigBee profile stack. In applications that perform complex and very repetitive processing, the usage of the microprocessor might become excessive. A general solution may be to use external memory, an additional microprocessor, or a controller, but this usually causes the cost and device size to increase.

We have shown that wireless sensor nodes can feasibly borrow the memory or computational resource from the gateway or the server in our preliminary study [5]. In this paper, we propose a resource management protocol (RMP) that enables wireless sensor nodes to efficiently use the resources including the memory and the CPU in the gateway or the server. Also, the effectiveness of the RMP is validated by several experiments through our implementation of the RMP.

The rest of this paper is organized as follows. Section 2 presents related works to efficient resource management in wireless sensor networks. Section 3 identifies the resource constraint issues and draws the requirements to relax them. In Section 4, we briefly introduce ZigBee-layered architecture. In Section 5, we propose an efficient resource management protocol. We evaluate the performance of our resource management protocol by experiments and analysis in Section 6 and make a conclusion in the last section.

2. Related Works

Wireless sensor nodes have been used for various applications in surveillance, environment and habitat monitoring, structural monitoring, healthcare, and disaster management [6]. Developers of wireless sensor nodes face technical challenges

that include dense ad-hoc deployment, dynamic topology, spatial distribution, and constrains in bandwidth, memory, computational resources, and energy [7]. Low-power sensor nodes are known for their limited resources. For instance, motes are equipped with kilobytes of RAM which may be easily insufficient for storing or processing images [8]. Typically, in visual sensor networks (VSNs) which consist of tiny visual sensor nodes called camera nodes, the camera nodes should be equipped with memories of larger capacity in order to store the data [9]. Using more powerful microcontrollers equipped with sufficient RAM for data processing would be a straightforward solution for large data processing [8]. But they usually cause the cost to increase. Traditionally, wireless sensor nodes collect data and transmit data to centrally resourceful gateway for processing because they are generally supposed to be resource constrained [8]. Memory overhead is one of the main technical concerns for sensor network security such as any replication detection protocol [10]. Existing solutions for multicast in wireless sensor networks are limited because they either support multicast only from a single source node (usually the root node) or they limit the multicast group size to constrain memory usage [11]. It becomes especially difficult to implement them when composing a large-scale wireless sensor network or controlling a peripheral device, such as an LCD, due to insufficient resources. This issue is described in Sections 3 and 6 in greater detail.

Toward efficient resource management in wireless sensor networks, several studies have been made on new protocols that are efficient in resource management and carefully managed by operating system level [12] and solutions to the scalability of resources using an open source cloud model which provides the storage and computation resources necessary to address the scalability [13]. Extension of the cloud computing paradigm to the sharing of sensor resources in wireless sensor networks results in a much promising technology called Sensor Clouds [14]. Since the resource and capability of physical sensor devices are limited, Sensor-Cloud infrastructure can be behalf of the sensor management such as availability and performance of physical sensors [15]. Various physical sensors with different owners can join Sensor-Cloud infrastructure. The templates for virtual sensors and virtual sensor groups are prepared for sharing physical sensors. Users can control their virtual sensors directly or via their Web browsers [15]. Mass data processing is required for these sensor networks. Several studies about employing virtual memory to increase the available memory have been made [16]. Virtual memory named FaTVM for data-intensive applications in wireless sensor networks makes it possible for sensor nodes to carry out complex computation with heavy memory footprint without using energy-hungry MPUs with large RAM [8]. FaTVM uses NAND flashes as secondary storage and focuses on reducing virtual memory overhead [8]. In our resource management protocol, wireless sensor nodes can borrow the resources from a server that has access to more sufficient resources in the network in such a way that is general and extensible enough to resolve the resource constraint issues in several kinds of wireless sensor networks.

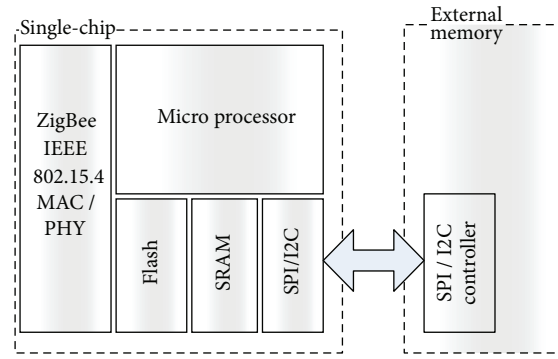


Figure 1: An Example of using an External Memory.

3. The Resource Constraint Issues

ZigBee-based wireless sensor nodes require low-power, low-cost wireless networking in residential and industrial environments [17], so that the battery in the sensor end node lasts for a long time without being plugged in. A busy state using complex computational resources consumes much more power than an idle state. Even memory resource constraints require additional memory devices, which increase the total system cost. To achieve low-power, low-cost wireless networking, the amount of memory allocated and the MPU usage in a device must be reduced. However, for large-scale sensor networks, with limited resources at every node, and deployment in environments with high access cost, the task of managing and operating these systems is extremely challenging [18].

Once a ZigBee end device (ZED) has joined a personal area network (PAN), a ZigBee coordinator (ZC) needs to update its neighbor table and store the minimum 12 bytes of information for the joined device [19]. If we assume that a maximum of 240 allowable devices have joined the network, the ZigBee Coordinator needs about 5 kilobytes of memory. In addition, ZigBee End Devices may require more memory to support the ad hoc on-demand distance vector (AODV) routing feature. Moreover, additional computational resources are required to access the memory. ZigBee end devices and the ZigBee coordinator may have various peripherals, depending on the application. For instance, in Smart Grids [20], a smart energy device that actively informs customers via in-home displays (IHD) of when or how energy is being used, has become necessary [21]. The device, which has an LCD screen, needs a great deal of screen buffer memory and computational resources to print text or to draw a picture on the screen.

To solve the resource constraint issues, a general solution is redesigning the hardware device. Though the software can be optimized, this cannot be a complete solution. Figure 1 shows an example of using external memory through a general-purpose interface, like SPI or I2C. The memory addition is simple, but it may require an API for interfacing external memory if the microprocessor or the single-chip does not provide a dedicated external memory interface. These APIs also use a substantial amount of resources.

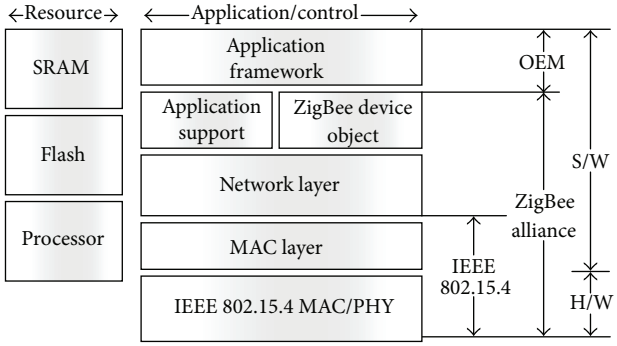


Figure 2: ZigBee-layered architecture.

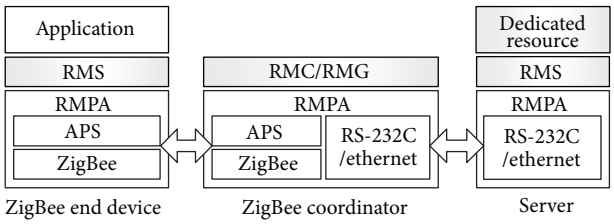


Figure 3: The resource management protocol-layered architecture.

In the same manner, we can use an additional processor or dedicated peripheral controller if more computational resources are required. However, this causes the device size to increase, and the total system cost becomes high.

4. ZigBee-Layered Architecture

ZigBee technology consists of application, network, medium access control (MAC), and physical layer. The MAC/PHY layer of ZigBee is specified in IEEE 802.15.4. ZigBee standardizes network and application layer. In other words, the specification scope of ZigBee alliance is the rest upper layer of IEEE 802.15.4 PHY/MAC. Figure 2 illustrates the ZigBee-layered architecture.

As shown in Figure 2, the application layer consists of an application support layer (APS), an application framework (APF), and a ZigBee device object (ZDO). The ZigBee alliance standardizes APS and ZDO. APF is handed over to a vendor. The network layer provides functionalities that enable an end device to communicate with other end devices. It also manages network security and routing.

4.1. Data Service. A network layer data entity (NLDE) sends and receives a data frame or controls a network header and communicates with APS using an NLDE service access point (NLDE-SAP). It also communicates with the MAC layer using a MAC common part sub-layer (MCPS-SAP) data interface.

4.2. Management Service. For the purpose of management, a network layer management entity (NLME) communicates with MAC using MLME-SAP. An application communicates with the network layer using ZDO.

Table 1: Header information block.

Layer	Header information
MAC	Length (1), MAC header (9), MAC CRC (2)
NETWORK	NWK Header (8), NWK security (14), NWK MIC (4)
APS	APS header (5~8), APS security (5), APS MIC (4)

4.3. Service Access Point (SAP). The data between two layers is transferred by SAP. Each SAP transfers an appropriate data structure, which is specified with an entity in the PAN information base (PIB).

4.4. Application Support Layer (APS). The maximum payload length in 802.15.4 is 128 bytes, including a length byte. Each layer uses an information header block, as shown in Table 1. As a result, the maximum payload length in the APS layer is 73 bytes. Moreover, it may be shortened further if the packet includes a routing header.

The ZigBee APS packet includes an application profile ID that describes the format of the message, a cluster ID for this message, a source endpoint, a destination endpoint, a bitmask of options, a group ID for this message if it is multicast mode, and a sequence number.

5. Efficient Resource Management

In a client-server system, a client's application may not be executed due to insufficient resources, even though the system has one or more resource-rich servers. The idea of efficient resource management is based on the imbalance of resources. The resource management protocol provides a way of using server-side resources.

Figure 3 illustrates the connection between the resource management client (RMC) and the resource management server (RMS). The resource management client requests a needed resource to execute a function for an application. The resource management server allocates a requested resource, performs a requested operation, and provides the outcome of the request to the resource management client. The ZigBee coordinator also acts as a resource management gateway (RMG) that relays packets between the resource management client and the resource management server. Each resource management client can request a resource from the resource management server through the resource management gateway. The resource management protocol adaptation (RMPA) layer abstracts the APS and ZigBee-based wireless sensor network interface.

5.1. RMP Packet. The format of packets that are transmitted from the resource management protocol is shown in Figure 4. The source address, SRC, is a short address of the server or the device that generates the RMP packet. The packet body includes an RMP command and an optional data field, as shown in Figure 4. The basic RMP command set

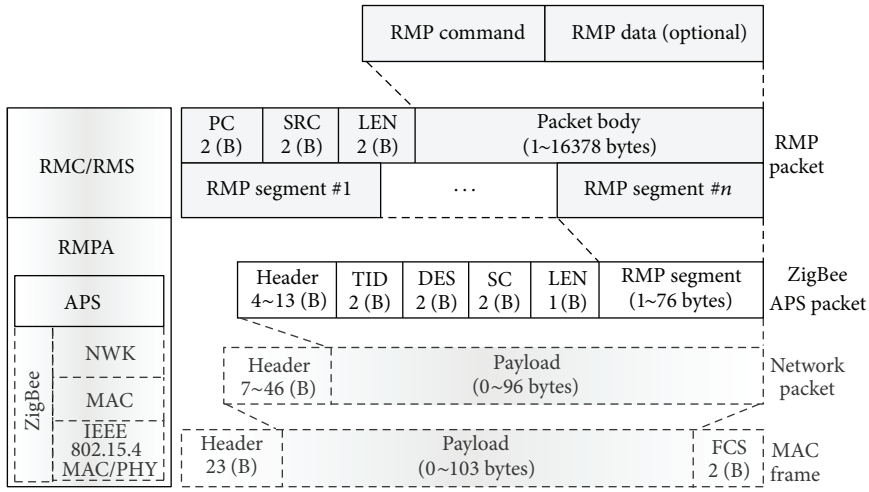


Figure 4: RMP packet structure in RMP layered architecture.

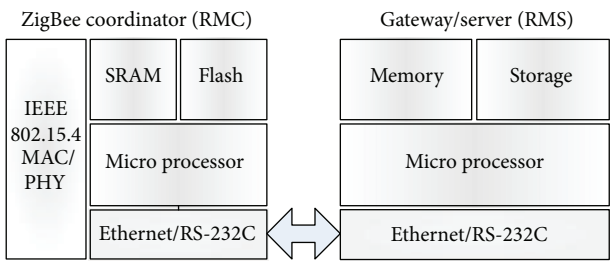


Figure 5: Data frame architecture.

Table 2: Packet types for solving issue on the large-scale ZigBee network.

Field	Description
STORE	Store data into the specified location of the RM server
LOAD	Load data from the specified location of the RM server
REMOVE	Remove data from the specified location of the RM server
FIND	Find a location of which is matched with specified condition from the RM server and load data

is shown in Table 2. It is also used for the experiment in this paper. The RMP command set can be modified or extended to conform to the requirements of other applications.

5.2. ZigBee APS Packet. The RMP packet is divided into the RMP packet segments shown in Figure 4. The APS header consists of the profile ID, the cluster ID, the source endpoint, and the destination endpoint. The transaction ID, TID, is a transient value that is specified by the initiator of the current RMP transaction when the original RMP packet is generated. The RMP packet segments that are generated by the RMP packet have the same TID. The destination address, DES, is a short address of the device that is the destination of the RMP segment. The sequence control, SC, has the number of the current segment and the number of the total segments.

5.3. Packet Concatenation. The maximum payload of a ZigBee bearer is 76 bytes, excluding the overhead of 12 bytes in the MAC layer, 26 bytes in network layer, and 13 bytes in APS layer, as shown in Table 1. Packet concatenation is needed at the destination, because a packet that exceeds the maximum payload size of 76 bytes in the resource management protocol is fragmented into several segments, which are sent to the destination. An RMP packet can be assembled by concatenating one or more raw data from the RMP packet segments that arrive at the destination. The RMP packet segments that have the same transaction ID are concatenated by the order of the sequence number.

5.4. RMP Packet Relay. The ZigBee coordinator between the resource management client and the resource management server functions not only as a resource management client but also as a resource management gateway, as shown in Figure 3. If the destination address of an APS packet that comes from the resource management server is different from the short address of the ZigBee Coordinator, the ZigBee Coordinator forwards an APS packet to the corresponding ZigBee end device without packet concatenation.

If the destination address of an APS packet that comes from the resource management client is undefined or identical to the short address of the resource management server, the ZigBee Coordinator forwards an APS packet to the resource management server. The ZigBee Coordinator receives an APS packet that includes the RMP payload and sends the APS packet to its destination address, as shown in Figure 4.

5.5. RMP Adaptation Layer. The RMP packet that comes from the resource management client or the resource management server forwards appropriate communications, which can be wireless (IEEE 802.15.4 RF) or wired (Ethernet, RS-232C) depending on the packet direction after regeneration of the packet header. In wireless communication, packet fragmentation can be applied to a packet that exceeds the

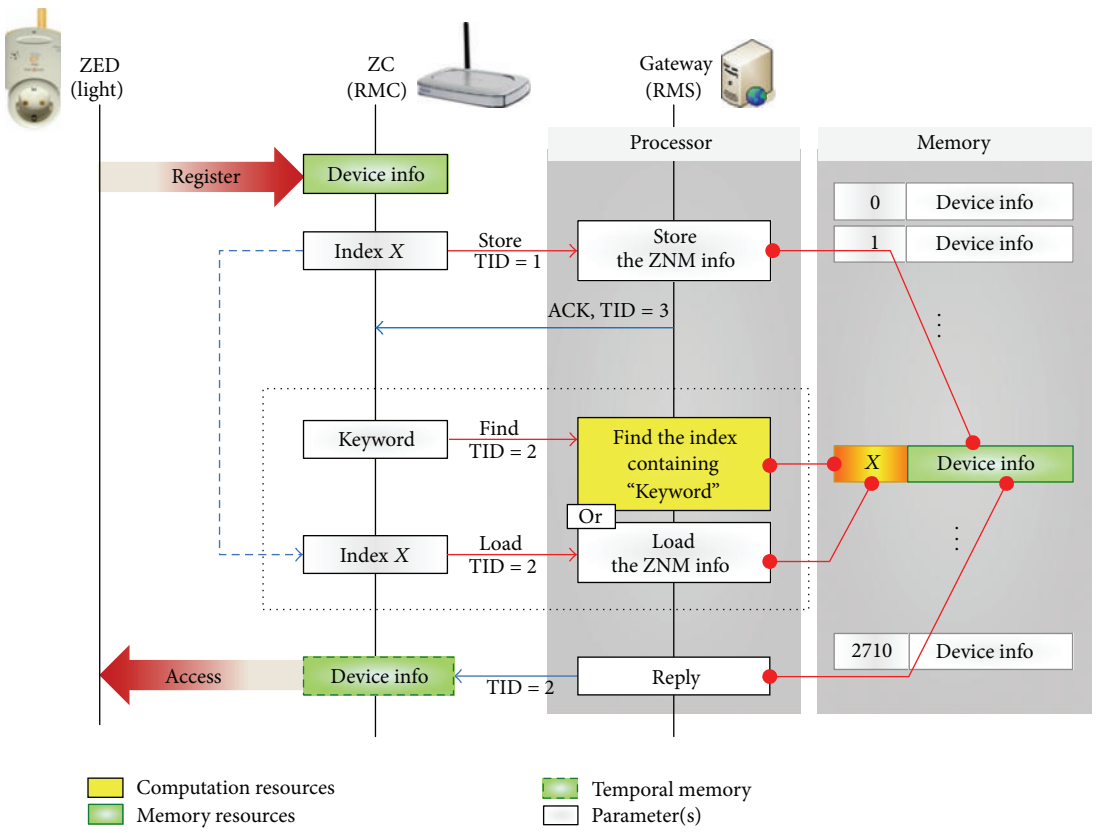
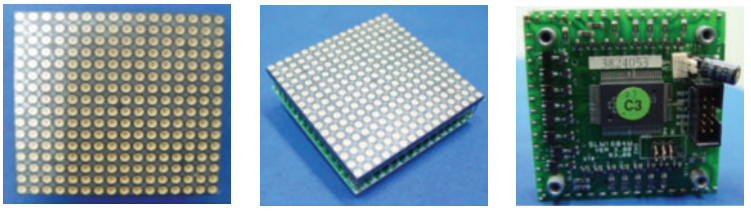


FIGURE 6: Sequence flow of the RMP on the large-scale ZigBee network.



(a) (b) (c)

FIGURE 7: 16 × 16 dot matrix.

maximum payload. The RMP adaptation layer concatenates the packet segments to the entire RMP packet prior to forwarding into the RMP layer. When transmission error occurs in the transaction, concatenation will fail and discard the packet segments if the packet segments arrive out of order. If the destination address of the RMP packet segment that arrived in the ZigBee Coordinator differs from the address of the ZigBee Coordinator, then the ZigBee Coordinator relays the RMP packet segment to the destination through the APS.

5.6. Packet Acknowledgment. Whenever the RMP packet is transmitted successfully, the resource management client or the resource management server has to reply with an acknowledgement within 2 seconds. If the retransmit time expires, then the sender starts the retransmit timeout procedures and retransmits the RMP packet using the same transaction ID.

5.7. RMP Types. Various packet types can be defined for different kinds of applications. Packet types can define various operations between the resource management server and the client, including memory allocation, memory access, data encoding, data decoding, and arithmetic operation. The RMP command and data shown in Figure 4 should be defined appropriately for an application.

6. Implementation & Experiments

We show actual implementations and experiments to validate the feasibility and effectiveness of the resource management

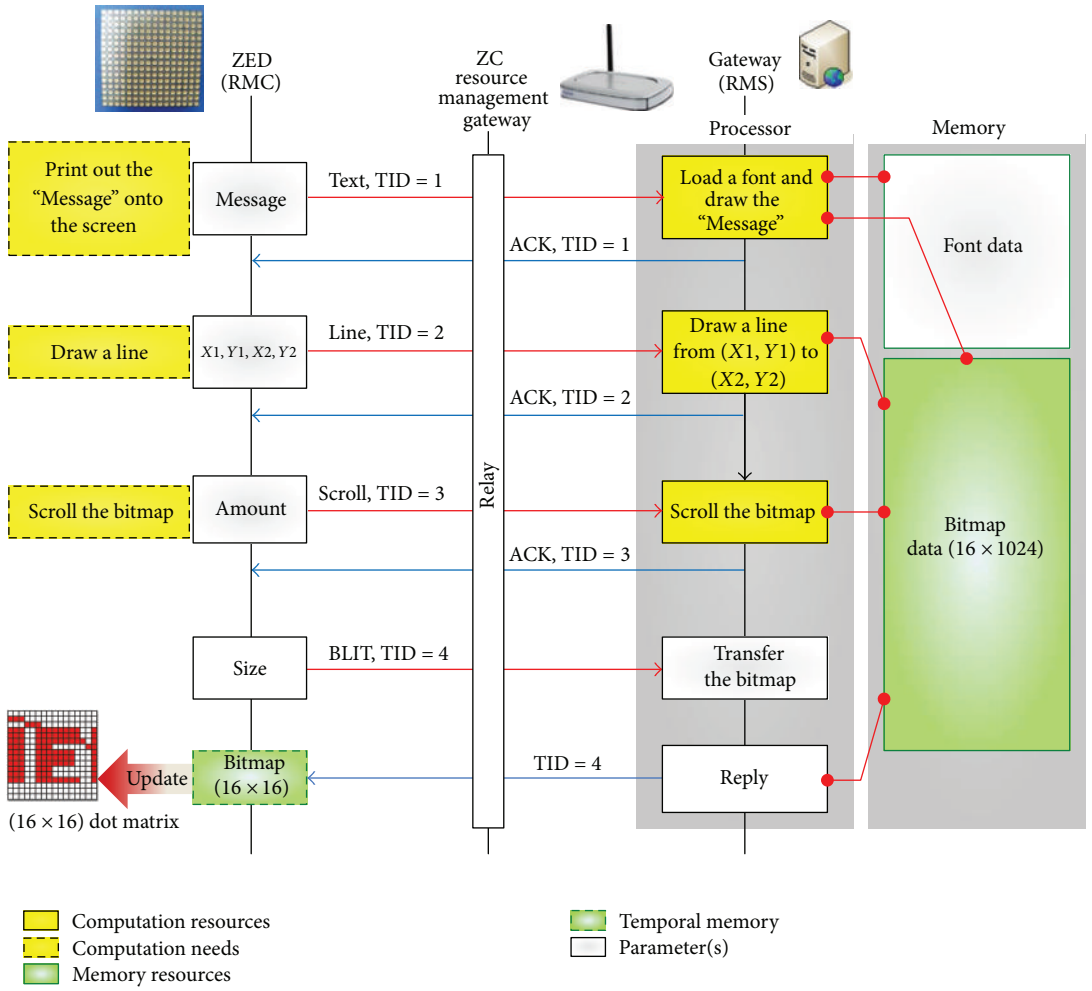


FIGURE 8: Sequence flow of the RMP for controlling a dot matrix.

protocol. Some packet types mainly used in wireless sensor networks with resource constraints are defined in Table 2.

6.1. The Resource Management Protocol on the Large-Scale ZigBee Network. In Figure 5, the ZigBee Coordinator acting as a resource management client typically connects with the resource management server via ethernet or an RS-232C cable. This means we can ignore the data transmission latency between the resource management client and the resource management server.

Figure 6 shows the sequence flow of the resource management protocol in a large-scale ZigBee network. The resource management client with insufficient memory sends an RMP packet with the packet type "STORE" and an index that represents the memory address in the server. The resource management server stores RMP data into its own memory with the specified index. When a resource management client needs to use the data, it sends a request packet with the packet type "LOAD" and the index. The resource management server then replies with the data associated with the index from its own memory. A packet with the packet type "FIND" enables the resource management client to look up the memory without using its own computational resources.

Now, we consider a building with 10,000 lights, each of which is connected and controlled by a ZigBee End Device in the ZigBee mesh network. They are controlled by the gateway, which is connected to the ZigBee Coordinator using RS-232C. During the peak time, an average of 1,000 control commands occur. The response time of the service T_s is 50 ms, which was obtained through experimentation. Referring to Little's Law,

$$\rho = \lambda T_s = 0.05 * \frac{1,000}{60} = 0.83 = 83\%, \quad (1)$$

where ρ is the server utilization and λ is the service requests per second

$$T_r = \frac{T_s}{1 - \rho} = 0.29 \text{ sec}, \quad (2)$$

where T_r is the response time for a service.

Because this shows that the server utilization ρ is less than 1 (100%) and the response time of 290 ms is acceptable

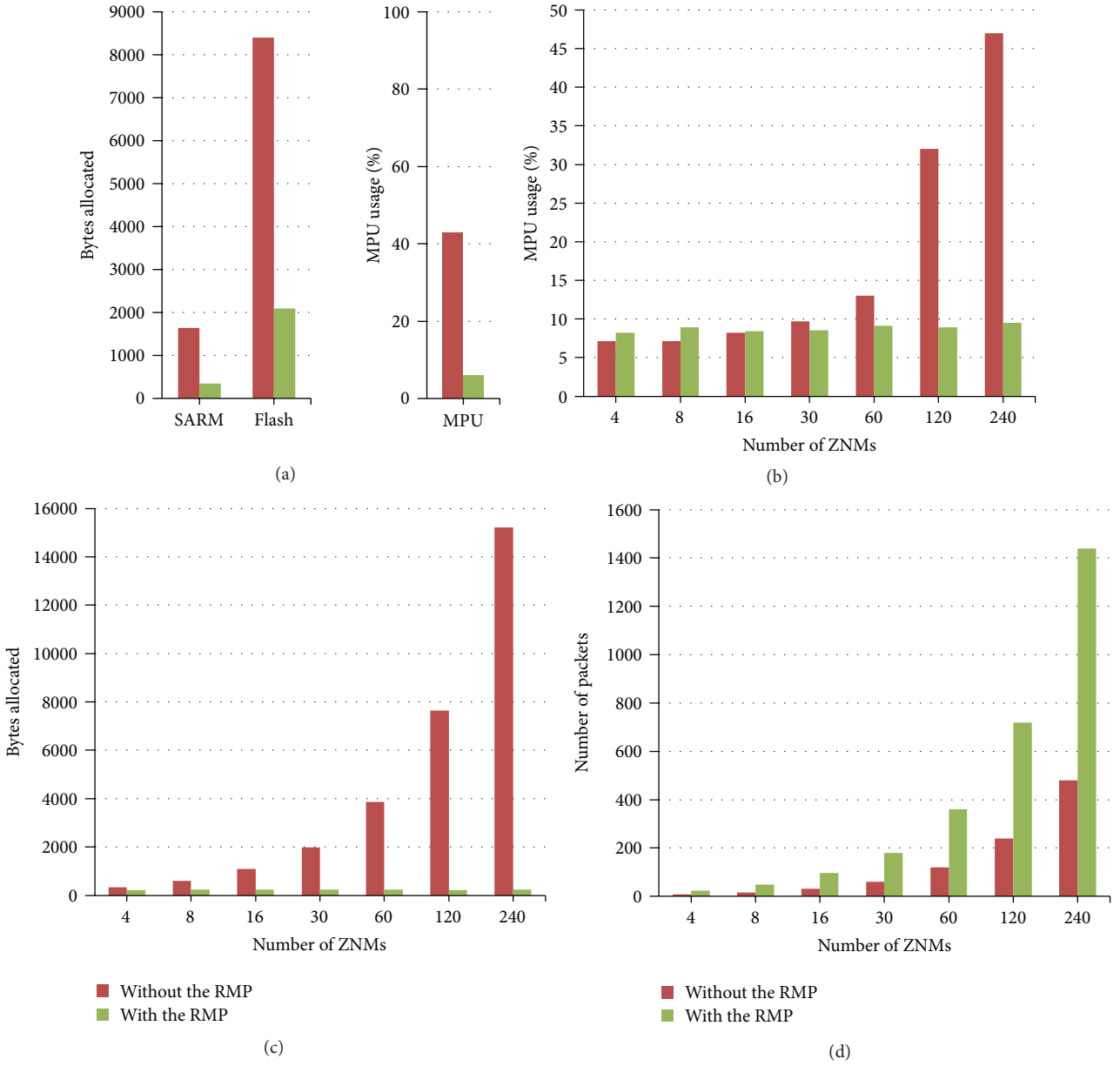


Figure 9: Performance of the RMP.

for a light control service, the assumption is reasonable and practical.

6.2. Controlling a Dot Matrix with the Resource Management Protocol. In order to validate the feasibility and the effectiveness of the resource management protocol for the resource constraints found in controlling peripheral devices, we set the ZigBee End Device as an in-home display using a 16 × 16 dot-matrix which has a 32-byte display buffer. Figure 7 shows an actual 16 × 16 dot matrix display used in the In-Home Display implementation. We define the RMP packet types as shown in Table 3.

A long text message with a maximum of 128 characters in several lines is then scrolled on the dot-matrix. For this application, the resource management server creates a 16 × 1024 (16 KB) drawing buffer for the 128-character-long

Table 3: Packet types for solving issues of controlling a dot matrix as a peripheral device.

Field	Description
FILL	Fill the specified region of bitmap data with a certain data
BLIT	Transfer the bitmap data to the RM client
TEXT	Print out a string to the specified location of bitmap space
SCROLL	Scroll the text or image
LINE	Draw a line onto the bitmap space of the RM server

message with an 8 × 16 font. Figure 8 shows the sequence diagram of the resource management protocol for controlling a dot-matrix. The resource management client sends a request packet which has the packet type "TEXT" and a message string to print out onto the dot-matrix. Then, the resource management server loads font data from its own memory and draws the message string onto the drawing buffer. The resource management client sends "LINE" request packets to the resource management server to draw lines onto the drawing buffer. Every 100 ms, the screen buffer is scrolled with the "SCROLL" request. When the server receives a "BLIT" request from the resource management client, its front partial 16 × 16 bitmap is transferred to the resource management client. The 16 × 16 bitmap is copied to the dot-matrix to update the display.

6.3. Experiments. For an experiment, we assembled one ZigBee Coordinator and four ZigBee End Devices to simulate 240 ZigBee End Devices. When each ZigBee End Device joins the network, a maximum of 60 ZigBee End Devices are assumed to join. Consequently, the ZigBee Coordinator stores the information 60 times with different indices. The lights are then randomly toggled by the gateway using a poisson distribution with a parameter of 100 controls per minute. We build two kinds of ZigBee End Device firmware. One uses the resource management protocol, but the other does not. They are written to the ZigBee End Devices evenly. The allocated SRAM size and the MPU usage, which are associated with controlling a dot-matrix and light, are measured from the kernel process scheduler of each ZigBee End Device. The varying numbers of the ZigBee End Devices are also measured from the ZigBee Coordinator. The needed flash memory size is evaluated with the firmware size. Figure 9 shows the result of the experiment.

As shown in Figure 9(a), by using the resource management protocol, the required memory size and MPU usage of the ZigBee End Devices that work as ZigBee Node modules (ZNMs) are largely reduced. Figures 9(b) and 9(c) show the effectiveness of the RMP for reducing the needed memory and MPU usage in the ZigBee Coordinator Module (ZCM). In Figure 9(d), when using the resource management protocol, the network traffic increases proportionally to the memory size that ZNMs need, because the ZCM converts the needed memory into RMP packets.

7. Conclusion

We have proposed a solution that enables efficient resource management of wireless sensor nodes using a resource management protocol. We have shown that it is feasible and effective for overcoming the resource constraints found in ZigBee applications through our implementations and experiments. The resource constraint issues were solved using the resource management protocol, without increasing the total system cost.

Disclosure

A preliminary version of this paper appears in Proceedings of the IEEE International Conference on Consumer Electronics, ICCE, 2011. This is the full version.

Acknowledgments

This research was supported by a Grant (SS100020) from the Seoul R&BD program funded by the Seoul Development Institute of the Korean government and supported by the Ministry of Knowledge Economy (10041725), Republic of Korea.

References

[1] W. C. Craig, "Wireless Control That Simply Works," ZigBee Alliance, 2004.

[2] W. Kluge, F. Poegel, H. Roller et al., "A fully integrated 2.4-GHz IEEE 802.15.4-compliant transceiver for ZigBee applications," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 12, pp. 2767–2774, 2006.

[3] Ember Corporation, "EM250 Single-Chip ZigBee/802.15.4 Solution," 120-0082-000R, May 2009.

[4] Atmel Corporation, "8bit AVR Microcontroller with Low Power 2.4 GHz Transceiver for ZigBee and IEEE 802.15.4," 8266AS-MCU Wireless, December 2009.

[5] W. H. Cho, J. Kim, and O. Song, "An efficient resource management protocol for handling small resource ZigBee devices," in *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE '11)*, pp. 765–766, January 2011.

[6] K. Römer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, 2004.

[7] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: a brief survey," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 41, no. 2, pp. 262–267, 2011.

[8] N. Lin, Y. Dong, and D. Lu, "Providing virtual memory support for sensor networks with mass data processing," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 324641, 20 pages, 2013.

[9] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, Article ID 640386, 21 pages, 2009.

[10] M. Zhang, V. Khanapure, S. Chen, and X. Xiao, "Memory efficient protocols for detecting node replication attacks in wireless sensor networks," in *Proceedings of the 17th IEEE International Conference on Network Protocols (ICNP '09)*, pp. 284–293, October 2009.

[11] A. Marchiori and Q. Han, "PIM-WSN: efficient multicast for IPv6 wireless sensor networks," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '11)*, 2011.

[12] P. Dutta and A. Dunkels, "Operating systems and network protocols for wireless sensor networks," *Philosophical Transactions A*, vol. 370, no. 1958, pp. 68–84, 2012.

[13] B. Sinha, *Wireless Sensor Network Architecture Addressing the Sclalability Issue Using Cloudsense*, SRM University, 2012.

[14] S. K. Dash, J. P. Sahoo, S. Mohapatra, and S. P. Pati, "Sensor-cloud assimilation of wireless sensor network and the cloud,"

in *Advances in Computer Science and Information Technology. Networks and Communications*, vol. 84 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 455–464, Springer, 2012.

[15] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure physical sensor management with virtualized sensors on cloud computing," in *Proceedings of the 13th International Conference on Network-Based Information Systems (NBiS '10)*, pp. 1–8, September 2010.

[16] N. Lin, Y. Dong, and D. Lu, "Fast transparent virtual memory for complex data processing in sensor networks," in *Proceedings of the International Conference on Sensor Networks (SENSORNETS '12)*, pp. 24–34, 2012.

[17] E. Callaway, P. Gorday, L. Hester et al., "Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 70–77, 2002.

[18] C. C. Han, R. Kumar, R. Shea, and M. Srivastava, "Sensor network software update management: a survey," *International Journal of Network Management*, vol. 15, no. 4, pp. 283–294, 2005.

[19] ZigBee Alliance, "ZigBee Specification," Document 053474r17, January 2008.

[20] B. Heile, "Smart grids for green communications," *IEEE Wireless Communications*, vol. 17, no. 3, pp. 4–6, 2010.

[21] ZigBee Alliance, "ZigBee Smart Energy Profile Specification," ZigBee Document 075356r14, May 2008.