pISSN 1976-913X
eISSN 2092-805X

J Inf Process Syst, Vol.9, No.1, March 2013 http://dx.doi.org/10.3745/JIPS.2013.9.1.041

A Dynamic Zigbee Protocol for Reducing Power Consumption

Do-keun Kwon*, Ki hyun Chung** and Kyunghee Choi***

Abstract—One of the obstacles preventing the Zigbee protocol from being widely used is the excessive power consumption of Zigbee devices in low bandwidth and low power requirement applications. This paper proposes a protocol that resolves the power efficiency problem. The proposed protocol reduces the power consumption of Zigbee devices in beacon-enabled networks without increasing the time taken by Zigbee peripherals to communicate with their coordinator. The proposed protocol utilizes a beacon control mechanism called a "sleep pattern," which is updated based on the previous event statistics. It determines exactly when Zigbee peripherals wake up or sleep. A simulation of the proposed protocol using realistic parameters and an experiment using commercial products yielded similar results, demonstrating that the protocol may be a solution to reduce the power consumption of Zigbee devices

Keywords—Zigbee, Low Power Protocol, Beacon, Power Consumption

1. INTRODUCTION

Zigbee [1] has been popularly adopted as a handy protocol for the equipment that requires low power consumption and low bandwidth. In many applications, Zigbee uses beacon packets to establish networks, which are called beacon-enabled networks. In a beacon-enabled network, which consists of a coordinator and peripherals, the coordinator periodically broadcasts beacon packets to the peripherals. Each beacon packet contains the information for network configuration, and a part of beacon has data to let the peripherals know the instants they have to wake up. Zigbee peripherals that usually have small batteries consume considerable power to handle the beacons. The power consumption shortens the life cycle of batteries and causes cumbersome maintenance work for exchanging batteries to make Zigbee peripherals work.

Several trials have been proposed to reduce the power consumption of sensor network device in the previous studies. One of the easiest ways is to make beacon interval longer. As the interval gets longer, Zigbee peripherals wake up less frequently, and they consume less power and stay alive longer without exchanging batteries. However, a longer beacon interval lengthens the time a Zigbee peripheral communicates with other Zigbee devices (like its coordinator). Sometimes a slower data exchange time brings more troubles than high power consumption. For example, Zigbee peripherals that are used for a fire alarm or as intrusion detection have to let the

Manuscript received January 25, 2012; accepted May 1, 2012.
Corresponding Author: Do-keun Kwon
* The Attached Institute of ETRI (kwondk@ensec.re.kr)
** Dept. of Electronics Engineering, Ajou University (khchung@ajou.ac.kr)
*** Dept. of Computer Engineering, Ajou University (khchoi@ajou.ac.kr)

coordinator know what has happened to it as soon as possible when it detects an emergency. The peripheral is required to send the event notification to its coordinator as quickly as it can. It's better to minimize the response time taken from the instant an event occurs in the peripheral to the instant the peripheral starts to send the event notification data to the coordinator. We will call the response time the *event transmission time* in this paper.

Meanwhile the role of a typical Zigbee coordinator is to broadcast beacons, to check whether its peripherals (connected to the coordinator through the Zigbee protocol) are alive or not, or to send application specific data to the peripherals. Thus, the data transfer time of a coordinator has to not be as fast as that of the peripherals in this case. In many cases, an external power source is usually provided to a coordinator and the coordinator needs to not consider power consumption as a major design factor. Thus, the problem of solving the trade-off between power consumption and data transfer time in many real Zigbee networks is converged as the problem of reducing the power consumption of Zigbee peripherals without lengthening the event transmission time.

The same problem frequently occurs in the sensor network. S-MAC[2][3] is one of popular protocols to solve the power problem in the sensor network. In S-MAC, peripherals check the channel and sleep periodically. Peripherals have to wake up to check a channel every period, even when they have no data to send or receive. Thus the waking up mechanism of S-MAC makes peripherals waste power unnecessarily. By extending the sleep period, S-MAC tries to reduce power consumption. But the extension introduces the increase in event transmission time. P-MAC[5] tries to resolve the problem of S-MAC somehow. P-MAC provides a mechanism to adjust the sleep interval, depending on the traffic in the network. Through dynamically controlling the sleep interval, P-MAC tries to reduce power consumption.

In this paper, we introduce a protocol to alleviate the problem of power consumption and event transmission time trade-off in Zigbee peripherals (not the coordinator) based on the similar philosophy (but a different approach) of P-MAC. In the proposed protocol, Zigbee peripherals wake up depending on their previous behaviors. When a peripheral had frequently woken up for exchanging data in the time period of T, the peripheral wakes up more frequently in the next period, T+1, assuming that it will have more data to take care of. However, the peripheral will only minimally wake up, if it had slept for long enough in the previous period.

Another observation we made is that many Zigbee peripherals are utilized in applications where the peripherals seldom detect emergency and need to quickly let their coordinator know about the events. In normal circumstances, it is usually enough for the peripherals to know who are alive and who are not. The best scenario for Zigbee peripherals to save power is that they sleep in normal circumstances and wake up quickly in emergencies. However, it is not possible to dynamically control the sleep pattern in a fashion such as with the typical beacon broadcast protocol. In this paper, a dynamic wake up control protocol is proposed to try to reach the best goal through a clever manner.

In Section 2, a dynamic Zigbee protocol for controlling the sleep pattern is proposed. The performance is evaluated through simulation and experiment in Section 3. The final section wraps up this paper.

2. A DYNAMIC ZIGBEE PROTOCOL FOR CONTROLLING THE SLEEP PATTERN

In this section, we propose a dynamic Zigbee protocol for saving power without slowing down the event transmission time in Zigbee peripherals. For dynamically controlling wake up

instances, the proposed protocol utilizes the "superframe" for allocating channels to Zigbee devices (coordinator and peripherals). The "superframe" is a frame format for Zigbee to synchronize devices in the beacon-enabled network. A superframe is connected by two beacons and Zigbee devices utilize the communication channel and exchange data during the time interval between the beacons, (i.e, a superframe). A superframe interval is divided into equal time slots, and one or more divided time slots may be allocated to a device.

The divided time slots are categorized into either the Contention Access Period (CAP) or the Contention Free Period (CFP). If a Zigbee device is allocated to the time slots in the CAP, the device has to compete with other devices to acquire the slots. But if a device is allocated to the slots in the CFP, the device uses the dedicated time slots safely. A slot in the CFP that is dedicated to a specific application requiring a fixed bandwidth is called a GTS (Guaranteed Time

Slot). A superframe may consist of a CFP and CAP. The beacon packet used in the superframe contains information about the time interval between two consecutive beacons, the start slot of the CFP, the addresses of devices assigned to specific GTS's, and other miscellaneous data. The proposed protocol uses superframes that only consist CFP slots. By utilizing the CFP, it becomes possible for the proposed protocol to control the instances and intervals when Zigbee devices wake up.

The coordinator in a network always wakes up and periodically broadcasts beacons and sends data down to the peripherals if possible. Thus we don't have to care about event transmission time to and from the coordinator. The power consumption of the coordinator is not our concern either since we assume that the coordinator has an external power source.

Meanwhile every peripheral wakes up whenever it needs to send data up to the coordinator. For example, if a peripheral detects a fire alarm, it immediately sends the information to the coordinator. That is, the event transmission time to the coordinator totally depends on the performance of the peripheral and is not our concern.

The concern of the proposed protocol here is to shorten the actual event transmission time from the coordinator to the peripherals while keeping the power consumption of the peripherals as small as possible. In the proposed protocol, peripheral D wakes up and sleeps depending on $S_D(T)$, which is called the *sleep pattern* of D, for the user defined time duration T, where $S_D(T)$ is defined as follows:

$$S_D(T) = S_{D,0}(T)S_{D,1}(T)\ldots S_{D,i}(T)\ldots S_{D,NF-1}(T)$$

where $S_{D,i}(T)$ is the sleep pattern of D for the i^{th} period of T, NF is the number of superframes in T, which is decided by considering network traffic, the number of peripherals (NP) in a network, and the bandwidth requirements. D sleeps during the i^{th} superframe if $S_{D,,i}(T)= 0$. Otherwise D wakes up in the superframe and checks the beacons. If the field in the beacons indicates that the data for D is set, then D receives the data during its own GTS of the i^{th} superframe. Otherwise, the device sleeps again. That is, $S_{D,i}(T)$ determines whether D has to sleep or wake up and receive data in the i^{th} superframe. Once $S_D(T)$ is determined (at very beginning of T), it controls the sleep pattern of D for the next NF superframes. $S_{j,0}(T)$ ($0\leq j<NP$) is set to "1" and never updated. That is, in the proposed protocol, Zigbee peripherals in network wake up during the first superframe of every T and sends their sleep patterns up to the coordinator. Broadcasting the sleep patterns allows the coordinator to know which peripherals wake up at which periods. Based on this knowledge, the coordinator decides on the time instants to sends data to or re-

ceives data from each peripheral, without making the peripherals wake up unnecessarily. The proposed protocol dynamically updates $S_D(T)$ to reduce the number of unnecessary wake-ups and the reduction consequently saves the power of Zigbee devices.

2.1 Updating the sleep pattern

All Zigbee devices in a beacon-enabled network have their own sleep patterns in the proposed protocol and the sleep patterns are updated every T period. The algorithm for updating the sleep pattern for D at time T, $S_D(T)$ is described in Algorithm 1. $S_D(T)$ is updated considering how many and how frequently events occurred in D during T-1. Here an event means a task that invokes D to exchange data with other Zigbee devices. $S_D(T\text{-}1)$ is updated to $S_D(T)$ at the first frame period of T, based on the statistics of behavior of D during T-1. Algorithm 1 describes the details.

Initially all bits of sleep patterns for all devices are set to "1." Setting the initial bit to "1" allows each Zigbee device to have a chance to receive the first superframes of other devices. If a device utilized one or more superframes and sent data (that is, one or more events have occurred) during T-1, it is assumed that the device still has data to take care of. And $S_D(T)$ is set to **111…11**, allowing D to be able to use all of the superframes during T. Meanwhile if the device woke up in the i^{th} period (due to "1" in $S_{D,i}(T\text{-}1)$), but there was no data the device had to handle, the number of consecutive "0's" in $S_D(T)$ increases to $2^{\boldsymbol{K}}$, where $\boldsymbol{K}$ is the maximum number of consecutive "0's"'s in $S_D(T\text{-}1)$. And the bit pattern of $S_D(T)$ becomes a series of repeated bits of ("1" and 2^K "0"). The bit series appears repeatedly until the number of bits reaches NF. For example, let $S_D(T\text{-}1)$ be "10010010." In this case, the maximum number of consecutive "0's," $\boldsymbol{K}$, is two. If there were no events occurring in the device but the device woke up, the number of consecutive "0's" increases to $2^{\boldsymbol{K}(=2)}$=4 in $S_D(T)$, and $S_D(T)$ becomes "10000100." If a larger $\boldsymbol{K}$ makes the number of consecutive "0's" exceed 7 (= NF -1 in this example), $S_D(T)$ becomes "10000000."

The peripheral P wakes up in the i^{th} period depending on the following three factors: $S_P(T)$ (P's sleep pattern), $S_C(T)$ (the coordinator's sleep pattern) and the occurrences of events that P has to handle in the period. When $S_{P,i}(T)$ = "1," the device wakes up, regardless of $S_C(T)$ and the occurrence of event. P wakes up when there are events P has to handle and $S_{C,i}(T)$ = "1" even if $S_{D,i}(T)$ = "0." The case allows P to handle emergency. that should be taken care of in that period.

if (*one or more events have occurred during the previous time interval T-1*) *then*
  $S_D(T)$ = ***111…11 ;***
or else if (*the device has woken up in one of periods in T-1, i.e,* $S_{D,i}(T\text{-}1)\neq 0$, *for some* $i > 0$)
then
{
  K = *maximum number of consecutive 0's in* $S_D(T\text{-}1)$;
  L=2**K;*
  Switch (*L*) {
    Case (***L*** < *NF-1*) : $S_D(T)$ = ***10….010….010…***
    Case (***L*** ≥ *NF -1)* : $S_D(T)$ = ***100……..…00***
  }
}
*or else /** $S_D(T\text{-}1)$ is equal to *1000000….00* and no event has occurred */
  $S_D(T) = S_D(T\text{-}1)$;

Algorithm. 1. Algorithm for updating sleep pattern

3. PERFORMANCE EVALUATION

We evaluate the performance of the proposed algorithm through both simulation and experiment. The performance is also compared with that of the typical beacon control protocol that utilizes fixed beacon intervals. We will call the typical protocol the "static beacon protocol." The evaluation focuses on how efficiently Zigbee peripherals save power without increasing event transmission time.

For the simulation, we assume that eight Zigbee peripherals are connected to the coordinator in a Zigbee network. The time interval for a superframe is set to 8 seconds. It means that it takes 8 seconds for a peripheral to receive the next beacon after it receives a beacon. We also assume that the Zigbee coordinator has an external power source. Thus the power consumption of coordinator is not something we were concerned about. We had the coordinator be always awake and broadcast beacons every 8 seconds in the simulation and experiment. First, the outcome of the simulation performed with precise actual parameters is presented. Then we address the experiment done with real Zigbee devices.

3.1 Simulation

To acquire the actual time and power consumption for the Zigbee devices (which will be used for the experiment) to spend in various states, they were measured in the experiment setting. Fig 1 illustrates the measured time and current consumption.

Zigbee devices stay in either the active or sleep state if (very short) transient states are ignored. The state where device *D* wakes up and exchanges data with other devices is called the *active state*. In the active state, the device monitors the channel and receives or sends data if needed. The time taken for an event transmission in active state is presented as (***S***-1) in Fig 1. If there is no data to handle in the active state, the device goes back to sleep and then goes into the inactive state. The time between waking-up and sleeping again is presented as (***S***-2) in Fig 1. In the case that the device sleeps for more than one superframe, the device wakes up, resets the timer at the starting point of every new superframe and then sleeps again. It takes a very short time and is presented as (***S***-3) in Fig 1. The actual measured times are summarized in Table 1. The table also includes the average current consumed in all states.

The time and current consumption by the proposed protocol with two different values of *NF* and the static beacon protocol with three different beacon intervals are compared. We named the proposed protocol with different *NF*'s (the beacon interval of 8 seconds) and the typical static beacon protocol with different beacon intervals as follows:

1) D1: Proposed protocol with $NF = 8$
2) D2: Proposed protocol with $NF = 16$
3) F1: Typical static protocol with beacon interval = 8 seconds
4) F2: Typical static protocol with beacon interval = 16 seconds
5) F3: Typical static protocol with beacon interval = 32 seconds

We assumed that events occur with Poison distribution in the Zigbee devices. The average time between the events varied from 1 second to 400 seconds. 100,000 events were generated in each case and each event evoked communication among the devices. The event occurrence mimicked the behaviors of devices that generate events.

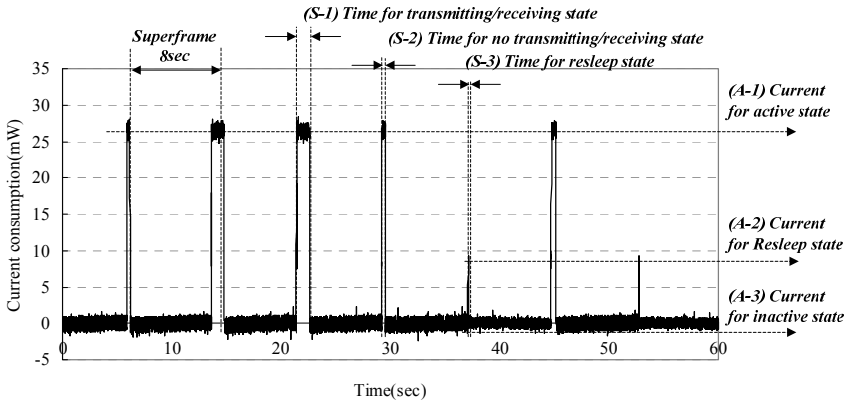

Fig. 1. Measured time and power consumption for simulation

Table 1. Time and current consumptions used in simulation

State / Consumption	S -1	S -2	S-3	Inactive state
Time consumption	1sec	0.27sec	0.01sec	NA
Current consumption	26.52mA	9.09mA	≈0mA	≈0mA

We measured the average power consumption per second as the first performance comparison metric. The total power consumption during the simulation was obtained by the product of the sum of the total current consumption and supplied power to the Zigbee device (= 3.3V). Dividing the total power consumption by the total simulation time yielded the average power consumption per second. That is, the average power consumption per second, $\boldsymbol{P_{avg}}$, can be presented as:

$$P_{avg} = \frac{3.3}{T_{sim}} \int_{0}^{t=Tsim} itdt$$

where T_{sim} is the simulation time, and i is the current consumption by a device during T_{sim}.

Fig. 2 shows the average power consumption of the simulated protocols. The protocols consume less power as events occur less frequently. It is obvious because as events occur less frequently, the devices wake up less frequently and thus consume less power. For the same reason, F3 consumes the least amount of power among F1, F2, and F3.

In the proposed protocol with different *NF*'s, (D1, D2), D2 consumes less power in all of the event intervals. This tells us that as *NF* is larger, the devices consume less power. This is because predicting $S(T)$ becomes more accurate as *NF* gets larger. That is, D2 with a larger *NF* forecasts the sleep patterns of devices more accurately and consequently reduces the number of unnecessary wake-ups than D1 does. The power consumption by the approaches of D1 and D2 occur less frequently than those of F2 and F3 as events.

The next simulation evaluates the event transmission time in the protocols. We will look at event transmission time from two viewpoints. One view is the event transmission time for sending a message from a coordinator to its peripheral and another view is from the peripheral to the

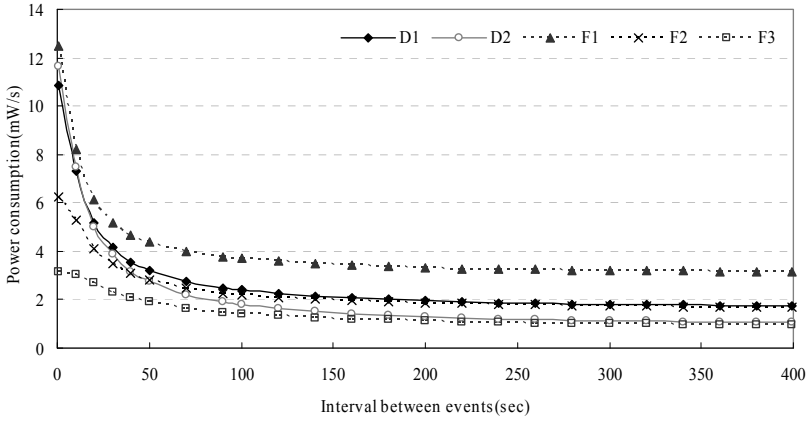

Fig. 2. Power consumption in different sleep control protocols

coordinator. The same set of events utilized for the power consumption simulation was used for simulating event transmission time.

3.1.1 Transmission time from peripheral to coordinator

The average event transmission time from a peripheral to the coordinator for the static protocol converges to half of the beacon intervals. This is because the randomly generated events, on average, are transmitted in the half of beacon interval. Since the coordinator is always awake (note that we assume that the coordinator is always awake since the coordinator with an external power source is free from consumption concerns), the peripherals send event occurrence data to the coordinator during their assigned CFP's whenever events occur. Thus, the event data will be sent in about half of the superframe period (beacon interval) on average, after an event occurs. For example, let us assume that the fourth (out of eight) CFP is assigned to a peripheral in a Zigbee network where the beacon interval is 8 seconds and eight equal CFP's are in a superframe. If an event occurs sometime in the middle of the second CFP, the peripheral will send the event data during the fourth CFP. Thus, it has to wait for the transmission for more than 1 second but for less than 2 seconds. But if the event occurs at the fifth CFP, the event cannot be transmitted during the superframe. The event will be sent in the fourth CFP of the next superframe. Therefore, the event transmission time will be 8 seconds maximum. Accumulatively, we can say that the average event transmission time for the proposed protocol becomes 4 seconds, which is half of the beacon interval.

The average event transmission time from a peripheral to the coordinator in the proposed algorithm also converged to the half of beacon interval in 4 seconds. When a peripheral is awake, the circumstance is equal to that of the typical static protocol with a beacon interval of 8 seconds. Meanwhile when an event occurs in a superframe in which the peripheral sleeps and the peripheral has to send the event data to the coordinator, it wakes up and will send data in one superframe interval since at least one CFP of superframe has been assigned to the peripheral. Thus, the average transmission time also converges to half of the beacon interval in this case.

3.1.2 Transmission time from the coordinator to the peripheral

The event transmission time from the coordinator to a peripheral in the static protocol be-

comes half of the beacon period on average. This is because the coordinator is always awake and the peripheral receives beacons every beacon interval and thus event data from the coordinator arrives to the peripheral between the instant the event occurs and in the last moment of the beacon interval. The mean of the two instants becomes half of the beacon interval.

However, the event transmission time from the coordinator to the peripherals in the proposed protocol is different. Even when the coordinator has to send down emergent data to one of its peripherals as soon as it needs to, the transmission cannot be completed until the peripheral is awake and ready to receive the data in its CFP. Otherwise, the trial cannot be successful. Thus, the event transmission time depends on the sleep patterns of peripherals. As the coordinator sends data to a peripheral more frequently, the sleep pattern will contain more "1" bits. Thus, the event transmission time becomes shorter as the peripheral wakes up frequently. When the coordinator has a lot of event data to send to a peripheral, the frequent events will update the sleep pattern of the peripheral to have all "1's," causing the device to always be awake. The situation is the same as that for the static protocol with the beacon interval of 8 seconds, making the event transmission time 4 seconds on average. Meanwhile the sleep pattern contains more "0's" as the coordinator produces events less frequently. The peripheral wakes up less frequently and it takes more time for the coordinator to send event data to the peripherals. If no events occur in the coordinator, the sleep pattern of the peripheral will be updated to have all "0's" except for the first bit. in this case, the peripheral wakes up every 32 seconds for the proposed protocol with *NF*=4 and every 64 seconds for the protocol with *NF*=8 seconds. Since the average event transmission time from the coordinator to the peripheral converges to half of the wake-up period, the trans-

Table 2. Average event transmission time in simulation

Direction / Model	From a peripheral to the coordinator	From the coordinator to a peripheral
D1	4.001 sec	See Fig 3
D2	4.001 sec	
F1	4.001 sec	4.003 sec
F2	8.009 sec	8.009 sec
F3	16.042 sec	16.042 sec

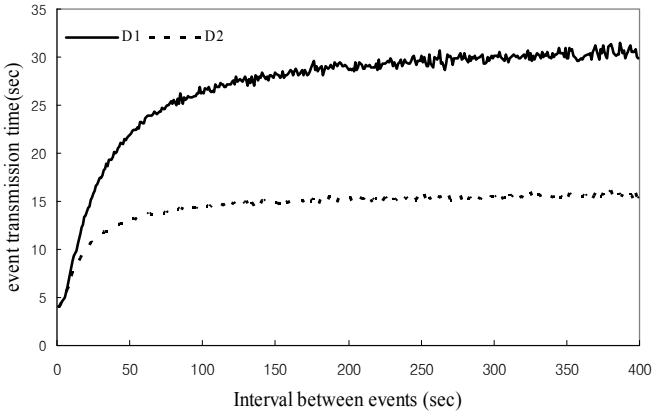

Fig. 3. Event transmission time in the proposed protocol

mission time becomes 16 seconds and 32 seconds for the proposed protocol, respectively. That is, the event transmission time from the coordinator to the peripheral becomes larger as *NF* becomes greater. Fig 3 shows the event transmission time from the coordinator to the peripheral.

3.2 Experiment

The performance was also evaluated in a real environment to see if the proposed protocol could actually overcome the obstacles introduced during implementation. The network was constructed with a coordinator and eight Zigbee peripherals. The coordinator and peripherals were products for home security. Six peripherals for door (or window) intrusion detection, one for the fire alarm, and one for gas detection were used for the experiment. The peripherals let the coordinator know the events as soon as they detected emergencies. Once the coordinator received an event notice, it sent an alarm to the home automation server, which was connected via the Internet, and then to the client through the phone line. The coordinator also broadcasted beacons and checked whether the peripherals were alive or not. Their hardware consisted of a popular processor, memory, a Zigbee transceiver, and I/O's. The application software for the Zigbee devices was implemented above the Zigbee network layer and Mac layer.

The event set used for the experiment was the same one that was used for the simulation. The current delivered to a Zigbee peripheral was measured with a high-speed data acquisition system that was connected to the peripheral through a small accurate resistor. The acquisition system sampled and measured the voltage drop across the resistor 100 times per second.

The experiment was done for 2 hours. Event transmission time and power consumption were measured when the interval between two consecutive event occurrences were set between 1 and 400 seconds, as was done in the simulation. But just for the sake of reducing the amount of measured data, we measured the event transmission time and power consumption when the interval was set to 1, 10, 20, 30, 60, 100, and 300 seconds, respectively.

The sampled current, I_{sample} became equal to $I_{sample}=V_{sam}/3.3$, where V_{sam} was the measured voltage at each sample instant. The average power consumption per second, P_{avg_sec} was calculated as:

$$\mathrm{V}\times(\sum_{i=1}^{f} I_{sam} / f)$$

where V is the source voltage of peripheral, 3.3V and f is the sampling frequency, 100Hz in the experiment. Finally, the average power consumption for the time during the experiment T, P_{avg} was calculated as:

$$\sum_{i=1}^{Tsim} P_{avg\_\sec} / T_{sim}$$

where T_{sim} is 7,200 (=2*3600) seconds in the experiment.

The average power consumption of the peripheral illustrated in Fig. 4 was nearly same as that in the simulation. As the interval between the events increases, the Zigbee device consumed less power in both the proposed and static protocols. The power consumption by the proposed protocol becomes close to the power consumption by the static protocol as events occur less frequent-

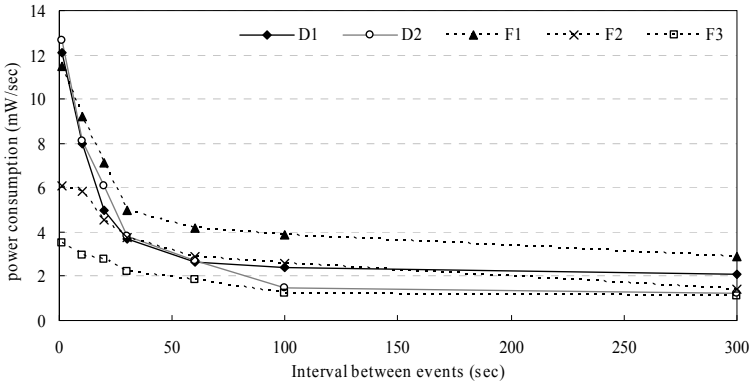

Fig. 4. Power consumption in the experiment

ly. Considering that events occur rarely in many real applications like the off-line home security application in this paper, we can say that the proposed protocol does not show a serious drawback in power saving. As *NF* becomes larger, the power consumption decreases more. The power consumption trend in the experiment was very close to that in the simulation in both the static and proposed protocol.

Table 3 summarizes the event transmission time that was measured in the experiment. The event transmission time measured in the experiment is also very close to that which was measured in the simulation. The event transmission time from a peripheral to the coordinator was around 4 seconds in the proposed protocol, which is comparable to that for the static protocol with the smallest beacon interval. In the static protocol, the event transmission time from a peripheral to the coordinator is close to half of the beacon interval.

The event transmission from the coordinator to the peripheral is also close to half of the beacon period due to the same reason as with the static protocol. However, the event transmission time varies with the frequency of event occurrence in the proposed algorithm. The transmission time is smaller when the event interval is smaller and the time increases as the event interval increases. Then the transmission time converges to half of the beacon interval, as the interval gets larger and larger. Fig. 5 shows how the transmission time changed. The outcome was close to that in the simulation.

The proposed protocol shows its superiority in both power consumption and event transmission time (from the peripheral to the coordinator) when few events occur and when the coordinator is always awake. Considering that many applications like home security need to only check their peripheral aliveness in normal circumstances, the proposed protocol may be a solu-

Table 3. Average event transmission time in simulation

Model \ Direction	From a peripheral to the coordinator	From the coordinator to a peripheral
D1	4.342 sec	See Fig 5
D2	4.189 sec	
F1	4.078 sec	4.172 sec
F2	8.193 sec	8.062 sec
F3	16.224 sec	16.348 sec

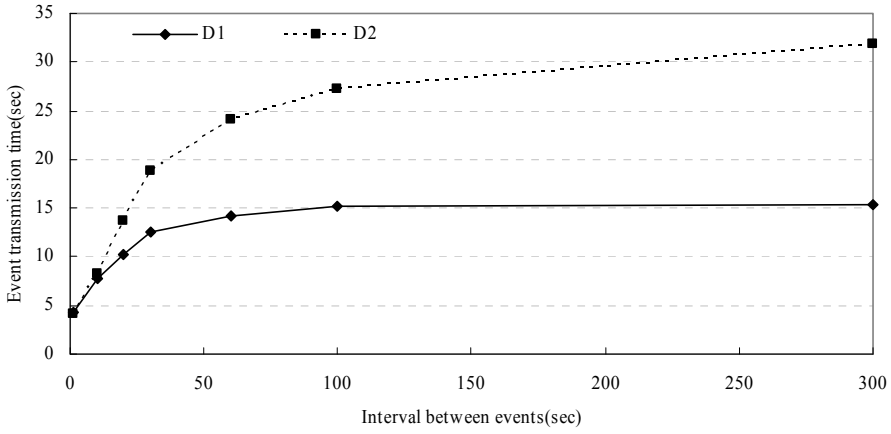

Fig. 5. Event transmission time in the experiment

tion to improve the trade-off between power consumption and the event transmission time (from the peripheral to the coordinator) for such applications.

4. CONCLUSION

This paper proposed a power saving protocol for Zigbee peripherals without delaying events being notified to the coordinator. The proposed protocol dynamically controls the wake-up times of Zigbee devices using the sleep pattern that has been updated depending on the previous event occurrence statistics. The power consumption and event transmission time from peripherals to the coordinator were simulated with real parameters and were measured in the experiment using commercial products. The event transmission time of the proposed protocol was very close to that of the typical static protocol with the smallest fixed beacon interval. But the power consumed by the peripheral in both the simulation and experiment approaches to the power consumed by the typical protocol with the largest fixed beacon interval as event occurs less frequently. The study demonstrates that it is feasible for the proposed protocol to save power consumption in Zigbee peripherals without increasing the event transmission time in many real applications.

REFERENCES

[1] "IEEE Standard for Information Technology-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)", Institute of Electrical and Electronics Engineers, 2003.

[2] Wei Ye, "Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks", IEEE/ACM transactions on networking, Vol.12, No.3, Junuary, 2004, pp.493-506.

[3] Wei Ye, John Heidemann, Deborah Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks", INFOCOM 2002, New York, Vol.3, Junuary, 2002, pp.1567-1576.

[4] T. V. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks", SenSys'03, Los Angeles, Noverbem, 2003, pp.171-180.

[5] Tao Zheng, Sridhar Radhakrishnan and Venkatesh Sarangan, "PMAC: An adaptive energy-efficient

MAC protocol for Wireless Sensor Networks", IEEE WMAN 05, April, 2005.
[6] Peng Lin, Chunming Qiao and Xin Wang, "Medium Access Control With A Dynamic Duty Cycle For Sensor Networks" , WCNC 2004 - IEEE Wireless Communications and Networking Conference, No.1, March, 2004, pp.1522-1527.
[7] ZigBee Alliance, www.zigbee.org
[8] Ashit Talukder, Rajankumar Bhatt, Tanwir Sheikh, Rishi Pidva, L Chandramouli , S. Monacos, "Dynamic Control and Power Management Algorithm For Continuous Wireless Monitoring in Sensor Networks", Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), Vol.00, November, 2004, pp.498-505.
[9] Ilker Demirkol, Cem Ersoy, and Fatih Alagoz, "MAC Protocols for Wireless Sensor Networks", IEEE Communications Magazine, (IN PRESS), 2005.
[10] Peng Lin, Chunming Qiao and Xin Wang, "Medium Access Control With A Dynamic Duty Cycle For Sensor Networks", Wireless Communications and Networking Conference, 2004, Vol.3, March, 2004, pp.1534-1539.
[11] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks", Proceedings of the Hawaii International Conference on Systems Sciences (HICSS), January, 2000.
[12] Alec Woo and David Culler, "A transmission control scheme for media access in sensor networks", Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, Rome, Italy, July, 2001, pp.221-235.
[13] Poole, I, "What exactly is … ZigBee?", Communication Engineer, IEEE Communications Engineer, Vol.2, No.4, August-September, 2004, pp.44-45.
[14] G. Lu, B. Krishnamachari, C.S. Raghavendra, "An adaptive energy efficient and low-latency MAC for data gathering in wireless sensor networks", Proceedings of 18th International Parallel and Distributed Processing Symposium, April, 2004, pp.224.

Do-Keun Kwon

He received the BS and MS degrees in Electronic Engineering from Ajou Univ. in 2004 and 2006, respectively. Currently, he is a researcher in The Attached Institute of ETRI, Republic of Korea. His research interests are in the area of embedded system and security.

Ki hyun Chung

He received a Ph.D. degree in Electronics Engineering from Purdue University in 1990. From 1991 to 1992, he was a researcher in Hyundai Semiconductor Research Center. Since 1992, he has joined in the department of Electronics Engineering at Ajou University in Republic of Korea.

Kyunghee Choi

He received a Diplome D'Ingenieur in the department of Information Engineering of ENSEEIHT, and a Ph.D. dgree in the department of Information Engineering of Paul Sabatier University in France in 1982. Since 1982, he has joined in the department of Computer Engineering at Ajou University in Republic of Korea.