An Energy-efficient Scheduling Framework for Multiple Bluetooth Low Energy Nodes

Jing-Ho Chen and Ya-Shu Chen, *Member, IEEE*

Abstract—Bluetooth Low Energy (BLE) is a wireless protocol that provides low-power communication for battery-driven devices. This protocol includes a connection parameter for extending the sleep time of BLE nodes. The protocol guarantees acceptable latency for sensing applications, partially by setting this connection parameter to pessimistic values; this can lead to excessive energy consumption. This paper presents a practicable energy-efficient scheduling framework for multiple BLE nodes with various data rates and latency constraints. We propose a connection parameter determination method and a blocking-aware scheduler for improving energy consumption and reducing blocking. The proposed methodology was evaluated through extensive experiments. The results revealed that the proposed algorithms prolonged the network lifetime by 170% compared with a simple pessimistic setting.

Index Terms—Bluetooth Low Energy, Real-time Scheduling, Power Management, Wireless Sensor Networks

1 Introduction

Power management in wireless devices is critical because such devices rely on limited energy supplies (i.e., batteries), and communication modules usually consume more power compared with the processors [1]. Because considerable demand has emerged for the Internet of Things, Bluetooth Low Energy (BLE) has been proposed to minimize energy consumption in sensor nodes. BLE enables designers to adjust the connection intervals of adjacent devices that exchange data through master and slave roles. The connection interval affects energy consumption and the latency of data communications [2]. Designers typically consider the usage parameters for the system and choose the connection interval parameter as a fixed value, such as 20 [illegible] for iOS [3] and SPP-over-BLE profile [4]. Short connection intervals increase energy consumption, but long connection intervals increase latency.

Real-time applications are increasingly deployed in wireless devices (e.g., health monitors, industrial control systems, and home security). Latency may be critical for certain applications; for example, a wearable health monitor must send abnormal heart rate information to its master device within a limited amount of time. Moreover, each wearable device might run multiple applications to monitor multiple types of information (e.g., electrocardiography, heartrate, and electromyography information). Different programs would have varying data rates and latency requirements (deadline constraints). The critical issue of how to minimize energy consumption with multiple latency constraints for several applications with various data rates has raised. A typical device network on the Internet of Things consists of multiple decentralized nodes, instead of a single node. In contrast to traditional wireless protocols, the BLE protocol does not support carrier sense multiple access with collision avoidance (CSMA/CA), because the individual physical channel is used for all nodes to conserve energy. When multiple BLE nodes are considered, the energy conservation challenge is complicated by the multiple node transmission scheduling problem and the latency constraints of various applications.

To conserve energy for multiple latency-sensitive BLE nodes, this study explores three technical problems: (1) the determination of the required connection interval for a sensor node, having multiple applications with varied data rates and latency constraints; (2) the assignment of a connection interval for each network node for energy minimization without violating latency constraints; and (3) the scheduling of node transmissions to resolve data collisions and meet latency constraints.

We propose an energy-efficient framework to solve these three problems; our framework prolongs the network lifetime and satisfies quality of service (latency) constraints. We also propose a demand packet estimation method for determining the required connection interval for each BLE slave without buffer overflows. We present an energy-efficient connection interval assignment that enables balancing the tradeoff between runtime blocking and energy consumption for a maximized network lifetime. We present a blocking-aware write-request scheduler that schedules node transmissions to maximize the quality of service. The evaluation results revealed that a considerable amount of energy can be conserved using this framework.

The remainder of this paper is organized as follows. Section 2 introduces the BLE protocol property in wireless sensor networks and a survey of the relevant studies. Section 3 presents the system model and the BLE node problem. In Section 4, we propose our energy-efficient scheduling framework for multiple BLE nodes, including the service rate and connection interval of each node. Section 5 provides our evaluation of the proposed approach in various situations. Finally, Section 6 offers a conclusion.

- *Ya-Shu Chen is with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei City, Taiwan, ROC. E-mail: yschen@mail.ntust.edu.tw*
- *Jing-Ho Chen was with National Taiwan University of Science and Technology.*

2 Background and Related Work

This section presents the properties of the BLE protocol and the technical challenges involved with these properties within the context of relevant studies.

2.1 The Connection Event in BLE Protocol

In a BLE network, nodes communicate through master-slave relationships. Each communication includes synchronization and a data exchange, and is called a *connection event*. These connection events occur periodically, and the period is determined by a *connection interval*. As shown in Fig. 1(a), when a connection event occurs, both master and slave nodes awake. The master sends a data request to the slave, and then the slave sends data and an acknowledgment to the master. After the transmission is complete, the BLE modes of both the master and slave nodes are set to sleep mode. When certain data are sensed by the slave-node processor, these data are stored in the slave buffer, to be transmitted during the next connection event. In other words, no data transmission can occur unless a connection event occurs. The data payload of each packet in the BLE protocol is 20 bytes. When the data buffer contains multiple packets, packets are sent in sequence during a single connection event.

Although longer connection intervals result in longer waiting times for data exchange, longer connection intervals also result in lower average currents (Fig. 1(b)), because long sleep durations consume little current. Most BLE nodes have a limited battery capacity, and thus, network administrators typically adjust connection intervals to maximize the lifetime of the network. This raises the issue of how to determine the optimal connection interval for a slave node for maximizing the lifetime of the network when providing an acceptable quality of service (for all packets).

2.2 Multiple Slave Nodes with Star Topology

In contrast to the original bluetooth protocol, the BLE protocol conserves energy by organizing data transmission among multiple nodes as a star topology, rather than as an ad hoc piconet (Fig. 2(a)). As shown in Fig. 2(b) [5], all slave nodes use a uniform physical channel with time division multiple access (TDMA) to communicate separately with the master. Therefore, each slave node can only wake up during its connection event; the slave node cannot remain awake to listen for incoming connections.

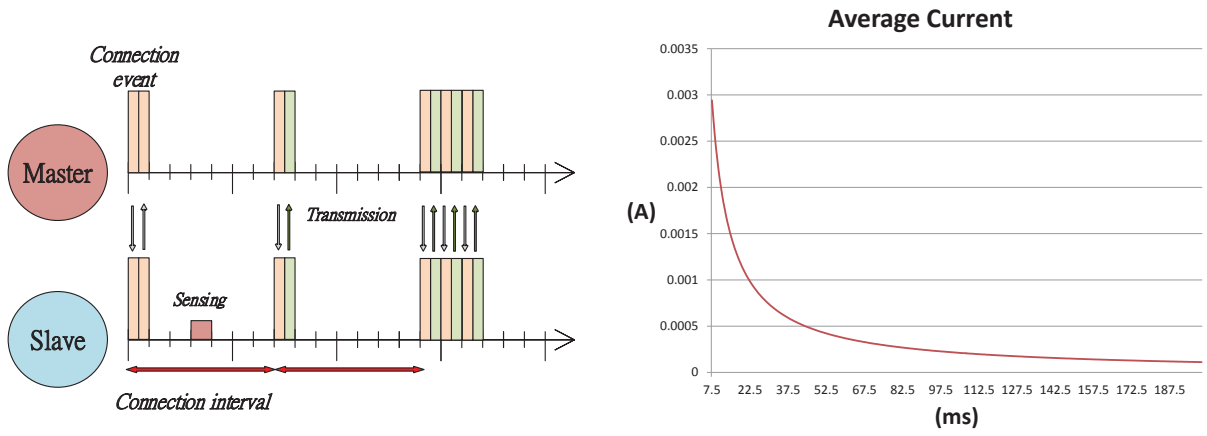
Because the BLE protocol is based on a master-slave relationship, the BLE protocol does not support CSMA/CA with a shared physical channel, and therefore, the collision problem [6], [7], [8], [9] can occur, as shown in Fig. 3(a); multiple packets are sent by a slave node $Slave_2$, but the slave node $Slave_1$ simultaneously begins its connection event and sends a packet to the master node. The packet sent by $Slave_2$ is dropped. To resolve this issue, a write-request idea was proposed in [10], according to which the master node assigns connection intervals to slave nodes and requests sensing data by writing from slaves, as shown in Fig. 3(b). This motivates researchers to determine appropriate connection intervals for multiple slave nodes, which prevents collisions and maximizes the lifetime of network.

2.3 Related Work

Numerous studies [11], [12], [13] have investigated energy efficient packet transmission problems in wireless sensor networks. In [11], the transmission rate was adjusted with the energy state and traffic load to minimize the packet delivery duration. In [12], the offline and online schedules for packet transmission were analyzed in terms of delays and energy performance levels. In [13], a low-complexity online schedule for packets balanced energy consumption and latency. Various real-time issues for wireless sensor networks (e.g., body sensor networks) are increasingly regarded as pivotal. Several studies [14], [15] have presented the real-time communication with earliest deadline first (EDF) scheduler in wireless sensor networks. Individual packet deadlines were studied in [16], which presented a channel-coding scheme for lowering the transmission power with deadline constraints. In [17], an optimal packet scheduling is proposed to guarantee the maximum throughput for the case where only two different deadlines are allowed. In [18], energy consumption was minimized with deadline constraints and Densest Interval First policy. In [19], the deadline miss ratio was analyzed in existing real-time scheduling algorithms under the Bluetooth protocol. In [20], event delivery latency and the event delivery ratio were used to satisfy latency requirements with multiple packets.

The BLE protocol has been proposed to minimize the energy consumption of sensor nodes. The BLE protocol differs from traditional WiFi and Zigbee protocols, because it allows packet exchanges between master and slave nodes only during a connection event, rather than at any time. The energy-efficient scheduling of BLE involves prioritizing the connection intervals of individual slave nodes, rather than packets. In [21], the power consumption levels for BLE, ZigBee, and Ant were compared under various data rates. In [22], throughput and power consumption were evaluated for data-streaming applications with various connection intervals and streaming packets, and the findings showed that the connection interval significantly effected the energy. In [10], a dynamic connection interval scaling framework was proposed for multiple applications with various data rates in a sensor node.

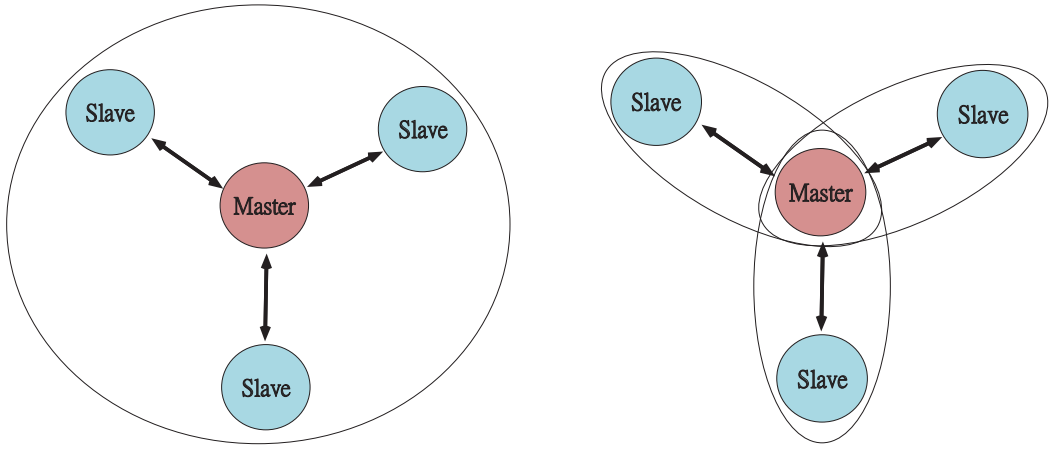
In contrast to single-node issues, packet collision issues in networks with multiple nodes are more critical. In [23], node-based and level-based scheduling was proposed through a distributed coloring of each node for constructing the TDMA table of each node. In [24], the authors proposed assigning each node an active time in an ultrawideband-based network by using a makespan scheduler. Regarding latency constraints, the authors in [25] supported the real-time queries of wireless cyber-physical systems, and proposed Real-Time Query Scheduling to satisfy real-time constraints without considering runtime packet collisions. In [26], authors proposed adaptive share polling to balance latency and power consumption. To minimize energy consumption, the authors in [27] proposed Low Duty cycle (LDC) and High Duty cycle (HDC) to serve the network and determine intervals between two successive channel polling for real-time traffic. In [28], authors proposed dynamically scaled polling intervals to conserve energy for multimedia traffic. In [29], authors proposed a fuzzy control system



(a) Connection Event

(b) Average Current

Fig. 1. Connection Interval



(a) Bluetooth Piconet

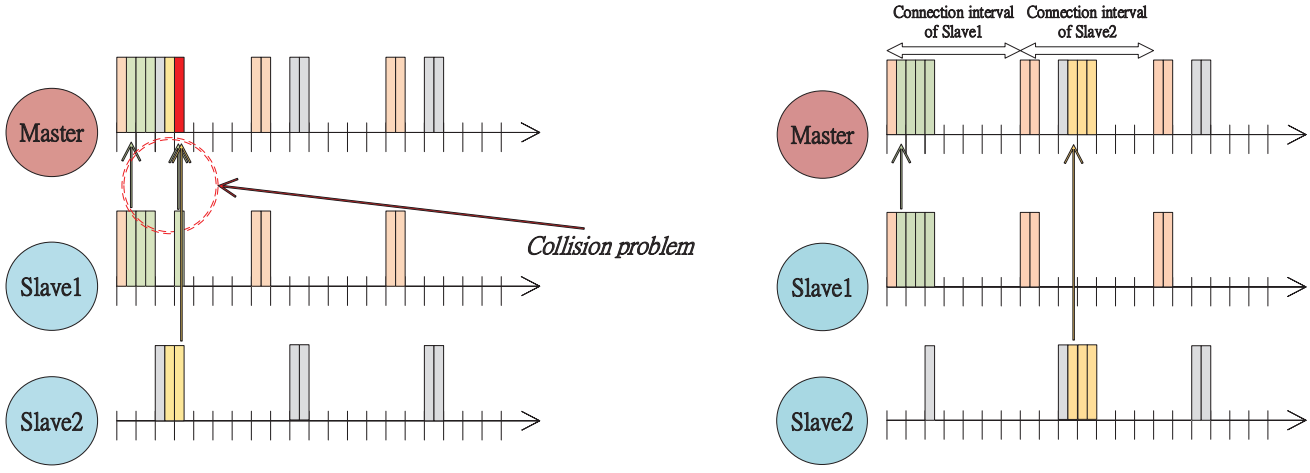
(b) BLE Star

Fig. 2. Bluetooth and BLE

for smart home power management that prolonged the lifetime of the network without real-time consideration. A priority polling schedule for prolonging the lifetime and reducing latency in health care applications is proposed in [30]. In [1], authors used a wake-one-radio to set the RX sniff duration; this satisfied latency constraints in a low-energy wireless sensor network. In [7], authors proposed analyzing the probability of collisions with various connection intervals. Even with different protocols in the same connection interval, a longer interval was found to lead to low probabilities of a collision. The present study examines the energy minimization issue for multiple BLE nodes with latency considerations.

3 System Model and Problem Formulation

This study explores an energy-efficient scheduling method to prolong the lifetime for a BLE network while maintaining the quality of service. With a set of BLE sensor nodes $N = \{n_1, n_2, ..., n_m\}$, each node n_i has multiple periodic sensing applications $E_{n_i} = \{e_1, e_2, ..., e_h\}$ and each application is denoted as $e_i = (\mu_i, p_i)$ where μ_i denotes the bytes of sensing data, and p_i denotes the sensing period. The lifetime of the set of sensor nodes is the duration from when nodes become active to when the battery of any node is exhausted. The lifetime $L(n_i)$ of each node n_i is defined by the ratio of battery capability C to the average current of the node I_{n_i}. The current of the node varies with the connection interval. Based on the specification of a BLE node [31], the average current of the node I_{n_i} can be estimated by the active current I_a, the active duration for a packet T_a, the sleep current I_s, and the connection interval C_{n_i} of a node n_i. When a single packet is considered, the current I_{n_i} of each node n_i



(a) Collision Issue

(b) Write-Request Scheduling

Fig. 3. Multiple Nodes Scheduling

can be regarded as $\frac{I_a \times T_a + I_s \times (C_{n_i} - T_a)}{C_{n_i}}$ [1]. Therefore, a longer connection interval can extend the lifetime of the network, but it might violate the latency constraints of the sensors. However, the relationship between connection intervals and current levels is not linear as shown in Fig. 1(b).

This study proposes determining connection intervals with a runtime scheduler that ensures the quality of service of the network and maximizes the lifetime of multiple BLE nodes. In this paper, we assume that the network has error-free links, packet losses in the link layer were not considered. If packet loss is considered, our approach can be extended with a loss rate, and details can be found in [32]. The present study proposes buffering all packets at the application level, and determining when all buffered packets are to be sent. We reserve the whole connection interval as the response duration for the packet scheduler in the link layer. The packet scheduler in the link layer can apply existing algorithms, such as First Come First Served with TDMA [33].

4 Energy-efficient Scheduling Framework for Multiple BLE Nodes

This section presents the proposed energy-efficient scheduling framework for multiple BLE nodes; service interval determination for multiple applications is discussed in Section 4.1; the connection interval assignment of each node is described in Section 4.2; and the runtime scheduler is explained in Section 4.3.

As show in Figure 4, each BLE slave node is added to the network, and all application properties (i.e., $e_i = (\mu_i, p_i)$) are sent to the master node. The master node gathers all information and determines the service interval while considering latency by using the **Multiple Event Interval** (MEI) method shown in Algorithm 1. The master node adjusts the connection interval to extend the network lifetime, and avoids online collisions by using the **Energy Efficiency Interval Multiple Access** (EIMA) method shown in Algorithm 2. After all connection intervals of each node have been determined, all the runtime packets of each slave node are scheduled using the **Shortest Interval First** (SIF) method shown in Algorithm 3 to maximize the quality of service by considering how non-preemptive data transmission might cause blocking.

1. When multiple packets are considered, the transmission current and the number of packets shall be taken into account as shown in Eq. 2.

4.1 Multiple-Event Energy Efficiency Interval

Algorithm 1 : MEI

Input: A set of applications $E_{n_i} = \{e_1, e_2, ..., e_j\}$, and each application e_i with the bytes of sensing data μ_i and the period p_i, the maximum data bytes of a packet δ, and the maximum sequence packets of an event Δ

Output: The service interval D

Sort E_{n_i} according to the period of each application in non-decreasing order
Set D as the minimal period in E_{n_i}.
while E_{n_i} is not null **do**
    Move the first element from E_{n_i} to E'_{n_i}
    Set the S as the maximum period in E'_{n_i}
    while $\sum_{\forall i \in E'_{n_i}} \lceil \frac{\mu_i}{\delta} \rceil \times \lceil \frac{S}{p_i} \rceil > \lfloor \frac{S}{D} \rfloor \times \Delta$ **do**
        $D = D - 1$
    end while
end while
Return D

To minimize the energy consumption of each node, the idea is to assign a service interval, which is the duration required for the slave to return its data to the master. It is convenient to provide the slave node with the most possible time. When considering only a single node with one software application, the connection interval of the node is equal to its required service interval. However, to provide

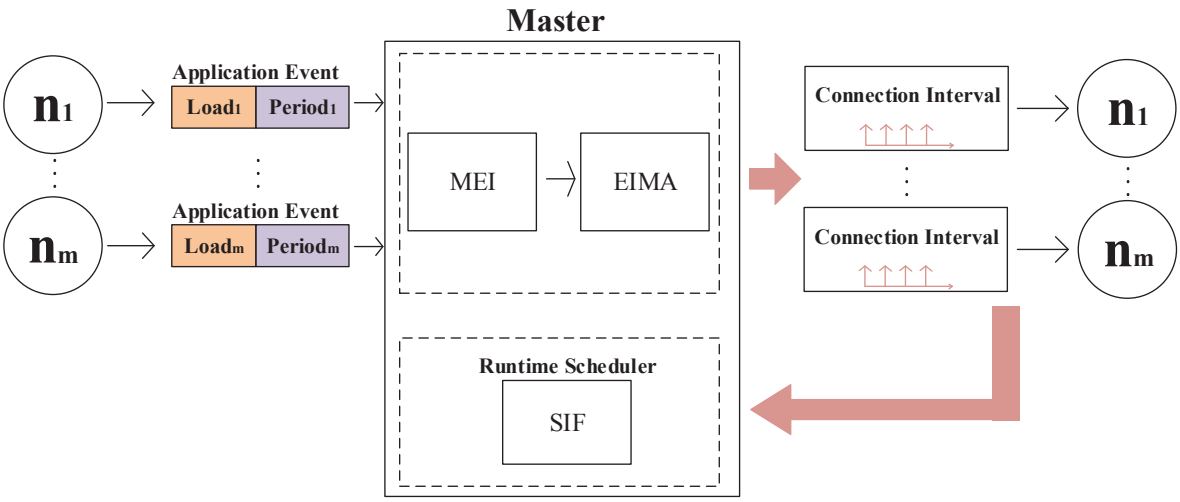


Fig. 4. The Energy-efficient Scheduling Framework for Multiple BLE Nodes

service to each application in a single node with several software programs, the separate demands of different programs must be estimated to assign a proper service interval. The details are shown in Algorithm 1.

With a set of periodic sensing applications $E_{n_i} = \{e_1, e_2, ..., e_h\}$, such that each application is denoted as $e_i = (\mu_i, p_i)$ where μ_i denotes the bytes of sensing data, and p_i denotes the sensing period, we sort all applications according to their periods, in non-increasing order. To meet the latency constraints of all applications, the service interval must not be shorter than the minimal period of all applications (Steps 1-2). In this study, multiple applications with various sensing periods (sensing rate) were considered. When multiple periods are considered, the supplied number of packets of the given service interval (i.e., $\lfloor \frac{S}{D} \rfloor \times \Delta$, where Δ represents the maximum sequential packets of an event) must be no less than all the sensing packets of all the applications (i.e, $\sum_{\forall i \in E'_{n_i}} \lceil \frac{\mu_i}{\delta} \rceil \times \lceil \frac{S}{p_i} \rceil$, where δ is the maximum data bytes of a packet) within time interval S. Because the maximum sequence packets Δ and maximum data bytes of a packet δ are limited, all required packets of each application can be estimated individually. The algorithm estimates the required packet quantities by iterating through periods, from the shortest to the longest period. If the currently assigned service interval (D) is too large, then a proper service interval must be assigned according to the required packets (Steps 3-7). In steps 6-7, the service intervals decrease sequentially, because the service interval in the decision equation (step 6) is located by the floor operation. If the periods of the applications are harmonic, the time complexity of the **Multiple-Event Energy Efficiency Interval** is $\Omega(h)$, where h is the number of applications. Otherwise, the time complexity is $O(p_{min})$, where p_{min} is the minimal period of applications, but the exact time complexity can be reduced when certain periods are divisible. No buffer overflow is in our framework, because none of the service intervals are shorter than the minimum period.

4.2 Energy Efficiency Interval Multiple Access

Algorithm 2 : EIMA

Input: Given a set of sensor nodes $N = \{n_1, n_2, ..., n_m\}$ and each node n_i with its service interval D_{n_i}

Output: The connection interval of each node

1: **for** each node n_i in N **do**

2: Estimate the average required current of each node

$$W_{n_i} = \frac{(I_a \times T_a) + (D_{n_i} - T_a) \times I_s}{D_{n_i}} \quad (1)$$

3: $W_{sum} = W_{sum} + W_{n_i}$

4: **end for**

5: **for** each node n_i in N **do**

6: $C_{n_i} = D_{n_i} \times \frac{W_{n_i}}{W_{sum}}$

7: Assign C_{n_i} to node n_i

8: **end for**

After we assign the proper service interval to each node with multiple applications, the next technical question is on how to determine the connection interval to avoid data collisions (Fig 3(a)) when multiple slave nodes are considered. The master can communicate with one slave node while blocking all other nodes; if other nodes are not blocked, some packet loss might occur by collision. To provide a connection interval of each node with appropriate blocking, the intuitive approach is to assign each connection interval as a ratio of the number of nodes to the shortest service interval for all nodes. If this is performed, the waiting time of each slave node is shorter than the required service interval. For example, with three nodes and a shortest service interval of 60 ms, then the connection interval can be reassigned as 20 ms to meet the latency constraints of all applications. However, a shorter connection interval entails greater energy consumption, as shown in Fig 1(b). Without loss of generality, certain nodes might have applications with long periods; these nodes can be serviced by long connection intervals; this can extend the lifetime of these

nodes, but it might result in unacceptably long blocking times for other nodes.

In this section, we present a method that reassigns nodes' connection intervals according to the current levels required by these nodes and the average lifetime of a node in the network. The details are shown in Algorithm 2. With the service interval of a node as estimated in Algorithm 1, we measure the required average current of that node by solving Eq. (1), and then reassign its connection interval by multiplying the ratio of its required average current to the summation of the required average currents for all nodes. When the connection interval gap is limited, the connection interval setting can be revised as $C_{n_i} = \lfloor \frac{D_{n_i} \times \frac{W_{n_i}}{W_{sum}}}{\omega} \rfloor \times \omega$, where ω is the minimal connection interval gap (e.g., $1.25\ ms$).

The idea behind Algorithm 2 can be explained with Fig 5; with the service intervals of three nodes, we can have the estimated current weights W_1, W_2 and W_3. Node n_3, which has the longest service interval, has the lowest current W_3. The assigned connection interval of n_3 is considerable shorter than its service interval to minimize the blocking times of nodes n_1 and n_2. However, to minimize the required current for extending the lifetime of the system, the connection interval of node n_1, which has the shortest service interval, is closer to its service interval than to that of node n_3. The time complexity of **Energy Efficiency Interval Multiple Access** is $O(m)$, where m is the number of nodes.

4.3 Shortest Interval First Scheduler

This section presents a runtime scheduler that guarantees the quality of service of the system. This scheduler forbids certain scheduling events to prevent collisions and synchronization problems. The scheduler implements a write-request model, in which the master node schedules the times at which it must request data from slave nodes, and each slave node is assigned a specific connection interval. The slave nodes conserve energy by only waking up to connect. Each slave node has only one connection interval during which it can transmit data, namely the connection interval authorized by the master node for that particular slave node. When the master requests data from any particular slave node, all other slave nodes are blocked. In a previous section, we proposed EIMA, which assigns connection intervals to minimize the blocking time for each node. An EDF scheduler can schedule nodes to guarantee the quality of service, but EDF might cause relatively long blocking times for nodes with relatively short deadlines if the nodes do not have their packets ready. The SIF scheduler, however, chooses nodes with the shortest deadlines, regardless of whether they are ready. The details of our proposed SIF scheduler are expressed as Algorithm 3.

When the system is initiated, and whenever any new node is added to the system, the MEI (Algorithm 1) is used to estimate each service interval required to accommodate all of the packets on each node. The master node sets the service interval of each node as the relative deadline of each node with multiple packets to be scheduled. Afterward, EIMA (Algorithm 2) is used to calculate the potential interference from multiple nodes, and to set the connection interval of each node. During runtime, each slave node wakes up to receive the connection parameters (during its connection interval)(Steps 2-8). The initial scheduling time is set as 0. The scheduler first finds the unique slave node that is ready with the shortest absolute deadline (n_j) and the node with the shortest absolute deadline, regardless of arrival time (n_k) (Steps 10-12). If the latest time for the last connection event before the deadline of node n_k ($\lfloor \frac{d_{n_k}}{C_{n_k}} \rfloor \times C_{n_k}$) is less than the blocking time caused by node n_j ($t + C_{n_j}$), to meet the latency constraint of node n_k, the master node remains idle without any requests until node n_k is ready, whereupon the master node resets the scheduling time to be equal to the arrival time of node n_k (Steps 13-19). Otherwise, the master requests the chosen slave node (n_j) with the shortest absolute deadline. Afterward, the master updates the absolute deadline and arrival time of the corresponding node by increasing it with the required service interval. The master node also updates the next scheduling time by increasing the current time (t) with the connection interval of the serviced node to avoid collisions and multiple node access problems. In other words, the master remains in the current physical channel of the connection event until all ready packets have been served or until the maximum packet number has been reached (Steps 20-24). The time complexity is $O(m)$ when the system is initialized, and $O(\log(m))$ otherwise, where m is the number of nodes.

Algorithm 3 : SIF

Input: A set of sensor nodes $N = \{n_1, n_2, ..., n_m\}$, each node n_i has multiple periodic sensing applications $E_{n_i} = \{e_1, e_2, ..., e_h\}$.

1: **while** The system is running **do**
2: **if** The system in initialized or a new node n_i is added into the system **then**
3: **for** each n_i in N **do**
4: Estimate the service interval D_{n_i} by MEI(E_{n_i})
5: $d_{n_i} = D_{n_i}$
6: $a_{n_i} = 0$
7: **end for**
8: Estimate connection interval of each node by EIMA(N)
9: Set t as the time unit with initial value 0
10: **end if**
11: Find the node n_j with $a_{n_j} \geq t$ and the shortest deadline d_{n_j} in N
12: Find the node n_k with the shortest deadline d_{n_k} in N
13: **if** $j \neq k$ **then**
14: **if** $\lfloor \frac{d_{n_k}}{C_{n_k}} \rfloor \times C_{n_k} \leq t + C_{n_j}$ **then**
15: Wait for n_k arrival
16: $t = a_{n_k}$
17: $j = k$
18: **end if**
19: **end if**
20: Write request to n_j for data transmission
21: $a_{n_j} = a_{n_j} + D_{n_j}$
22: $d_{n_j} = d_{n_j} + D_{n_j}$
23: $t = t + C_{n_j}$
24: **end while**

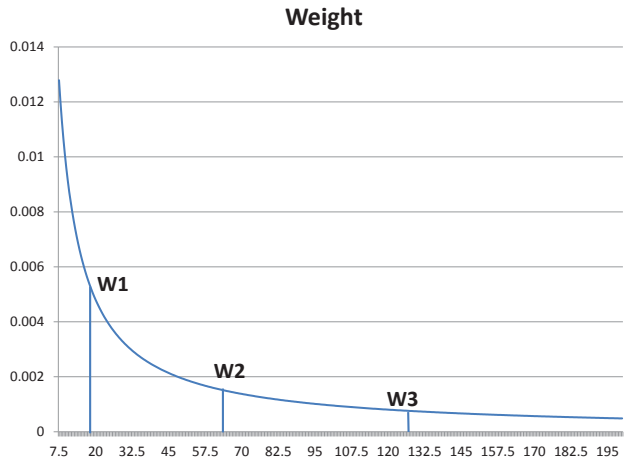


Fig. 5. EIMA Concept

4.4 Example

This section presents three BLE slave nodes n_1, n_2 and n_3 with different applications as an example of the effectiveness of the proposed framework. In node n_1, two applications, $e_1 = (10bytes, 100ms)$ and $e_2 = (10bytes, 200ms)$, exist. In node n_2, two applications, $e_1 = (10bytes, 500ms)$ and $e_2 = (30bytes, 500ms)$, exist. Node n_3 contains three applications, all of which have 10 $bytes$ of load and a 1000 ms period. Assume that the maximum number of packets Δ per connection event is 2, and the maximum bytes per packet δ is 20 bytes.

As shown in Algorithm 3, the service interval of each node is estimated by MEI (Algorithm 1) to guarantee the quality of service of all applications in all slave nodes. The service intervals of n_1, n_2, and n_3 are 100, 250 and 500 ms, respectively. When the greedy approach is applied, the service intervals of n_1, n_2, and n_3 are 100, 500 and 1000 ms, respectively. The required number of packets for node n_3 is $\lceil\frac{10}{20}\rceil \times \lceil\frac{1000ms}{1000ms}\rceil + \lceil\frac{10}{20}\rceil \times \lceil\frac{1000ms}{1000ms}\rceil + \lceil\frac{10}{20}\rceil \times \lceil\frac{1000ms}{1000ms}\rceil = 3$ when the service interval is 1000 ms. However, because the maximum number of packets in each connection event is 2, one packet cannot be transmitted within the connection event. This results in a failure to guarantee the quality of service.

After assigning the service interval of each node, EIMA (Algorithm 2) is used to estimate the connection interval by considering the access needs of multiple nodes. The current settings are based on the CC2541 chip [31], in which I_a is 8.246 mA, T_a is 2.675 ms, and I_s is 0.001 mA. The average required current levels for nodes n_1, n_2, and n_3 are 0.221 ($\frac{2.675\times8.246+(100-2.675)\times0.001}{100}$), 0.089 and 0.045, respectively. The connection interval of each node is then set as the product of the service interval and the weight ratio. The connection intervals of nodes n_1, n_2, and n_3 are 62.254 ($100 \times \frac{0.221}{0.335}$), 62.675 and 63.378, respectively. Under the BLE protocol, the minimal connection interval gap is 1.25 ms, and thus, the connection intervals of nodes n_1, n_2, and n_3 are 61.25 ($\lfloor\frac{62.2540}{1.25}\rfloor \times 1.25$), 62.5 and 62.5, respectively. If the greedy approach is applied, to meet the service interval constraint with three slave nodes accessing the network, the connection intervals of n_1, n_2, and n_3 can be all set to $\frac{100}{3}$ ms.

After a connection interval is assigned for each node, the proposed scheduler (SIF in Algorithm 3) schedules each node by calculating from the node with the shortest absolute deadline. The absolute deadline is derived from the service interval. Assume that all nodes are ready at time 0; the absolute deadlines of n_1, n_2, and n_3 are 100, 250 and 500 ms, respectively. The master node requests slave node n_1 first and updates the absolute deadline of node n_1 to be 200 ms. Afterward, the master waits for a response during the connection interval (i.e., 61.25 ms). At 61.25 ms, the master requests node n_2 and updates the absolute deadline of n_2 as 500 ms, and then waits for a response from n_2 within the connection interval (i.e., 62.5 ms). At 123.75 ms, the master requests n_1 again and updates the absolute deadline of n_1 as 300 ms. At 185 ms, the ready node with the shortest deadline is node n_3, and the node with the shortest deadline is node n_1. Node n_1 will be awake at 245 ms, and again at 306.25 ms based on the connection interval. If the master requests node n_3, the next scheduling time is 247.5 ms. Consequently, node n_1 will be requested at 306.25 ms, and all packets with deadlines of 300 ms will violate latency constraints. Based on SIF, $\lfloor\frac{300}{61.25}\rfloor \times 61.25 \leq (185 + 62.5)$, implies that node n_3 blocks node n_1, which has a shorter absolute deadline. Therefore, the master must not perform any action until 200 ms. The detailed schedule is shown in Fig 6.

We applied an RR method with a greedy connection interval setting to this problem, and the results and the detailed schedule are shown in Fig. 7. The Round Robin first schedules node n_1 before node n_2, after which it schedules node n_3. A comparison between Fig 6 and 7 shows that the response times in Fig 7 are better than those displayed in Fig 6. However, the required energy consumption in Fig 7 is nearly twice that in Fig 6 because of the shorter connection interval setting. A detailed performance comparison with energy consumption and the meet ratio is provided in Section 5.

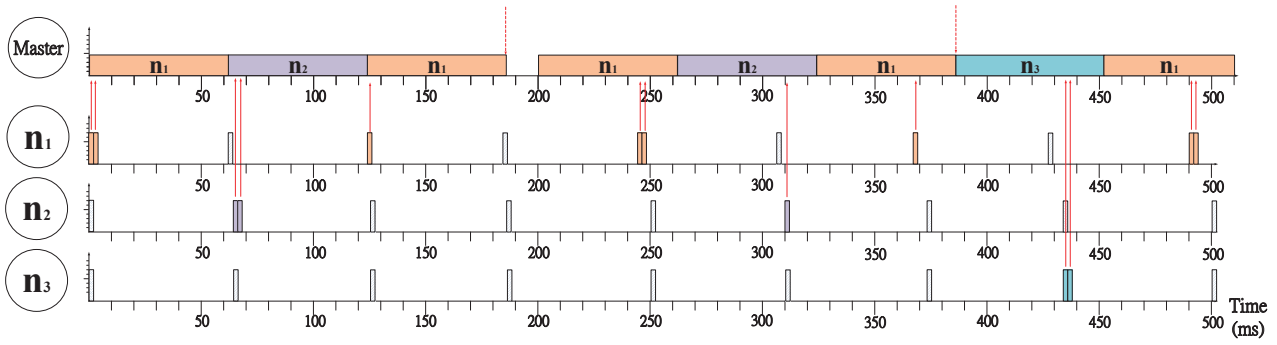


Fig. 6. EIMA with SIF Scheduler

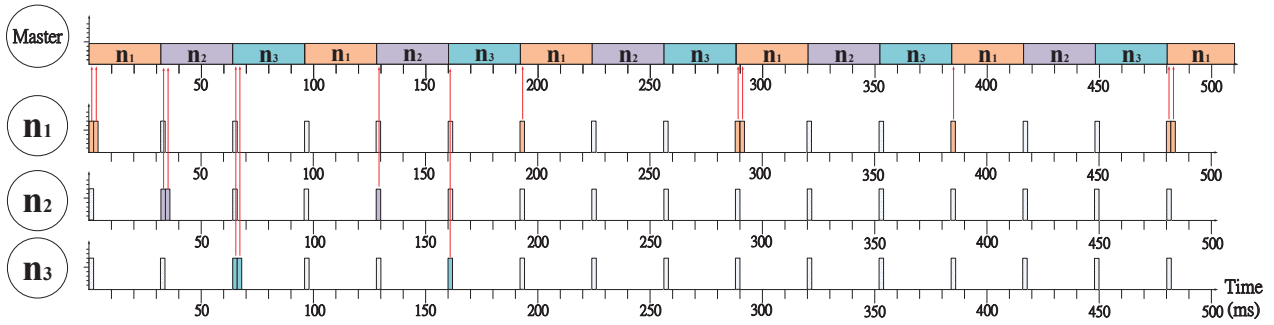


Fig. 7. Greedy with Round Robin Scheduler

5 Performance Evaluation

In this section, we evaluate three approaches for service interval determination in a single node (MEI), connection interval assignment for all nodes (EIMA), and runtime blocking-aware scheduling (SIF). We consider lifetimes and meet ratios with various data rates and several numbers of nodes. We describe the experimental setup in 5.1. Section 5.2 presents a comparison of the assigned service interval for a single node in each approach. Section 5.3 presents a comparison of the assigned connection intervals for multiple nodes in each approach. Section 5.4 and 5.5 offer comparisons on the schedules from each approach with various data rates and numbers of nodes, respectively.

5.1 Experimental Setup

This section presents the procedures involved in evaluating the lifetimes of networks and the meet ratios of applications with different approaches. The tests involve using various data rates for slave nodes. We compared our proposed MEI method with Lazy [10], Greedy, and iOS [3] methods. The Lazy method dynamically scales the service intervals by a callback function. This callback function gradually reduces the service interval as the difference between the maximum rate (R_{max}) and the minimum rate (R_{min}). The callback function operates only when the callback rate (R_{CB}) is smaller than the difference between the current rate (R_{cur}) and that between the maximum rate to minimum rate ($R_{max} - R_{min}$). With the lazy method, the current rate is estimated as $\frac{\delta \times \Delta}{C_{n_i}}$, where δ is the maximum number of packet bytes, Δ is the maximum number of packets in one connection event, and C_{n_i} is the current service interval. The maximum and minimum rates are calculated as the ratio of $\delta \times \Delta$ to the minimum period of all applications and the ratio of $\delta \times \Delta$ to the maximum period of all applications, respectively. The initial value of the callback rate is $200\ ms$ in this experiment. The built-in iOS method sets the service interval to the default value of $40\ ms$ for a single node.

For experiments with multiple nodes under a BLE protocol, we compared our proposed EIMA method with LDC [27] and iOS [3] methods. With LDC, the initial value of the service interval of each node is assigned by the MEI, and then the connection interval is estimated as the ratio of the initial value to the number of nodes. The built-in iOS method sets the connection interval to $20\ ms$ to serve multiple slave nodes [22].

Regarding the runtime scheduler performance, we compared our proposed SIF method to Polling [30], and Round

Robin [23] methods. Polling schedules the slave node with the shortest service interval first, and waits for its response until the connection interval is terminated. The Polling scheduler then schedules the node with the second-shortest service interval, and then the third-shortest, and so on. The loop repeats until all slave nodes are scheduled. The round robin schedules nodes sequentially according to its service interval, and repeats the scheduling in a loop.

In the following experiments, the workload is defined by a data rate varying from $80\ bytes$ to $360\ bytes$ [10], and each application period varies from $200\ ms$ to $800\ ms$ [14], [34]. Three slave nodes exist, each of which has two applications [35]; both applications are evaluated for each setting [2], [22]. The slave node settings are based on the CC2541 chip [31], in which I_a is $8.246\ mA$, T_a is $2.675\ ms$, and I_s is $0.001\ mA$. Based on these chip specifications, the connection current varies with the transmission current I_t. When multiple packets are transmitted, transmission current I_t is set to $9.564\ mA$, and the transmission time for each packet T_t is set to $1.82\ ms$; these settings reflect an awareness of the post-processing overhead [22]. Based on these parameters, the lifetime of each node n_i is estimated by the ratio of the battery capacity to the average current[2] $I_{n_i}^{avg}$, as shown in Eq. (2). For lifetime estimates in this experiment, the battery capacity is set to $230\ mAh$ [36]. In Eq. (2), the first term ($I_a \times T_a \times \lceil \frac{HP}{C_{n_i}} \rceil$) is the connection energy, the second term ($I_t \times T_t \times P_{n_i}^{HP}$) is the sequence transmission energy, and the third term ($I_s \times (HP - T_a \times \sum_{j \in n_i} \frac{HP}{p_j} - T_t \times P_{n_i}^{HP})$) is the sleep energy during the hyper-period (HP) of all applications in node n_i. The number of connection events can be estimated by $\frac{HP}{C_{n_i}}$, and the number of sequence transmissions ($P_{n_i}^{HP}$) can be calculated using Eq. (3). Because one packet can be transmitted by a connection event, the number of packet sequence transmissions is the difference between the total number of packets and that of connection events [22]. In Eq. (3), the first term is the total number of packets for all applications; this term can be estimated by the product of the total number of connection events and the required number of packets during the hyper-period. The second term is the number of connection events. After the average current of each node is estimated, the network lifetime is the minimum node lifetime; the node lifetime is the ratio of the battery capacity C to the average current of the node, as shown in Eq. (4).

$$I_{n_i}^{avg} = \{I_a \times T_a \times \lceil \frac{HP}{C_{n_i}} \rceil + I_t \times T_t \times P_{n_i}^{HP} + I_s \times (HP - T_a \times \sum_{j \in n_i} \frac{HP}{p_j} - T_t \times P_{n_i}^{HP}\} \times \frac{1}{HP} \quad (2)$$

$$P_{n_i}^{HP} = \sum_{j \in n_i} \frac{HP}{p_j} \times \lceil \frac{\mu_j}{\delta} \rceil - \lceil \frac{HP}{C_{n_i}} \rceil \quad (3)$$

$$L_N = \min\{L(n_i) = \frac{C}{I_{n_i}^{avg}}\}, \forall n_i \in N \quad (4)$$

2. According to the report by [1], the current consumed by data sensing is related much lower (about 8 times) than radio data transmission. In this paper, we focus on the energy consumed by BLE, and the energy consumed by data sensing is assuming as a part of the sleep current.

5.2 Connection Interval Setting with Multiple Packets

This section details our evaluation of the energy conservation and schedulability of our proposed MEI with multiple period applications on a single node. For a single node, the connection interval of the node is the same as the service interval.

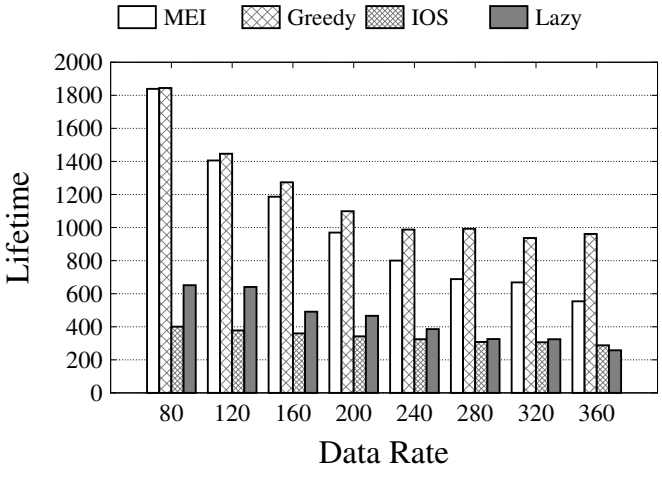
The horizontal axis in Fig. 8(a) indicates the varying data rate, whereas the vertical axis indicates the lifetime (hours). The lifetime of the proposed method (MEI) is near that of the Greedy method, with a lower data rate. The lifetime gap between the Greedy method and our MEI method increases with the data rate, because our MEI method shortens the connection interval to guarantee the quality of service for applications. The Greedy method assigns the minimum period as the connection interval without considering the data load, and thus, it produces a longer network lifetime compared with MEI. However, when the data rate is high, the schedulability of the Greedy method is lower compared with other approaches, as shown in Fig. 8(b). In contrast to the Greedy method, the Lazy method sets the shortest connection interval to serve applications when the maximum application rate is higher than the current rate, and then dynamically scales the service interval by using its callback function. The network lifetime of the Lazy method are shorter than those of the Greedy method, but the meet ratio is 100%. The lifetime of networks scheduled by iOS is the shortest among all network lifetime, because iOS sets the connection interval at $40\ ms$ for all data rates in a single node in order to achieve a high meet ratio. The lifetime of networks using MEI is higher than those of networks using iOS and Lazy methods, by up to 182% and 359%.

5.3 Connection Interval Setting with Multiple Nodes

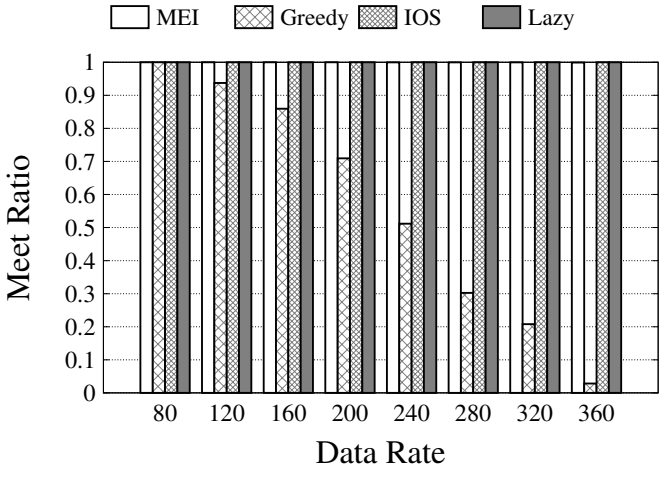
This section details our evaluation of the energy conservation capability of our proposed connection interval setting (EIMA) with multiple packets and multiple nodes. To improve schedulability, all approaches involved setting the service interval with MEI, and then they were scheduled using the proposed SIF method. More scheduler comparisons are presented in the next section.

As shown in Fig. 9(a) to lengthen the lifetime, EIMA assigns the connection interval to each node with the required average current consideration. EIMA is intended to set longer connection interval for nodes with shorter service interval, and to set shorter connection interval for nodes with longer service interval; this is intended to balance the lifetime and blocking time. The performance of the LDC method is worse compared with the EIMA method, because the LDC method tends to assign the same connection interval for each node, namely the assigned service interval divided by the number of nodes. This leads to shorter connection intervals. The networks planned by iOS have the shortest lifetimes among all of the approaches, because iOS sets the connection interval at $20\ ms$. The lifetime of EIMA is higher than those of iOS and LDC, by up to 235% and 54%.

As shown in Fig. 9(b), the meet ratio of the EIMA method decreases slightly when the data rate is low, because long service intervals produce long blocking times. Conversely, when the data rate is higher, the meet ratio of the EIMA

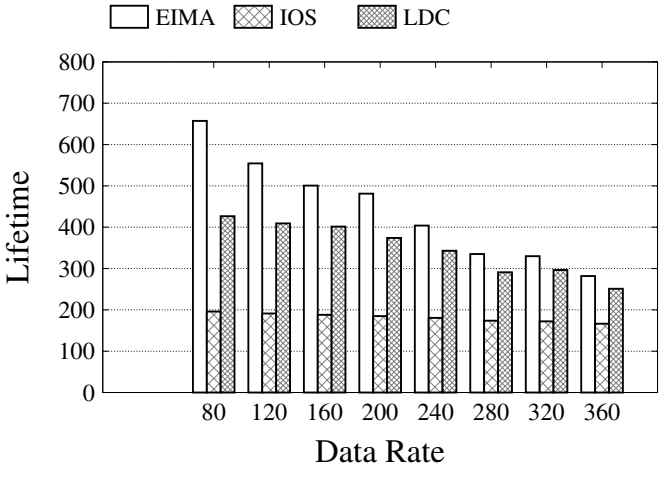


(a) Lifetime

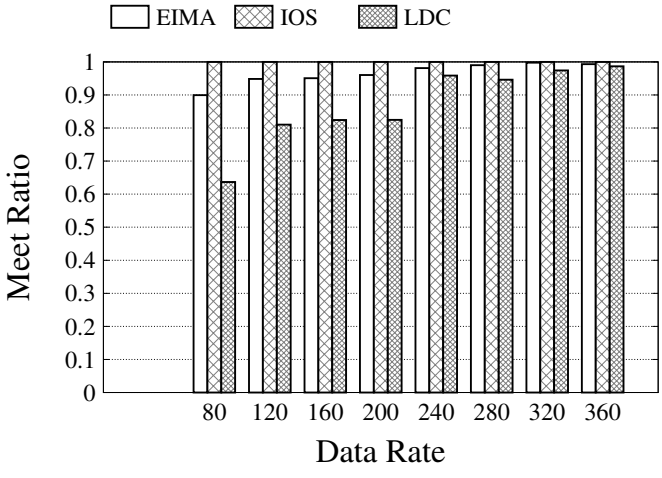


(b) Meet Ratio

Fig. 8. Varied Data Rate of Single Node



(a) Lifetime



(b) Meet Ratio

Fig. 9. Scaling Interval of Multiple Nodes

method increases because the connection intervals of nodes are similar and the blocking effect decreases. The meet ratio of the EIMA method is higher than that of the LDC method by up to 35%, showing that the EIMA method involves a superior tradeoff between energy and blocking.

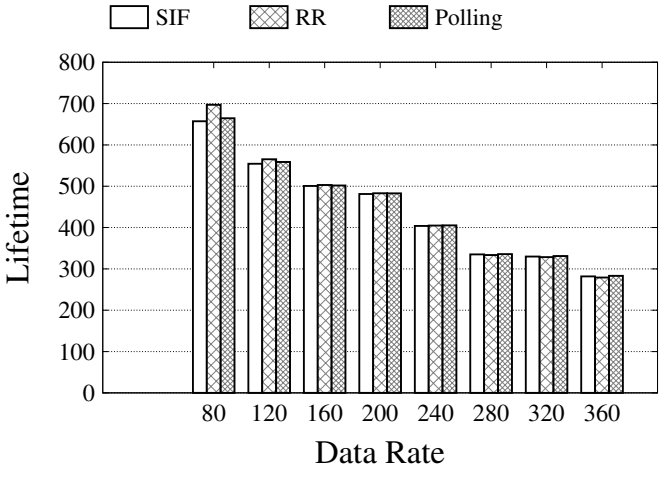
5.4 Runtime Scheduler

This section details our evaluation of the performance of the proposed runtime scheduler. The service and connection intervals of each node were set by MEI and EIMA, respectively. The lifetime of RR is slightly higher than those of others, as shown in Fig. 10(a). With RR , each node can send its packets in each round, resulting in fewer accumulated packets for each node. Therefore, the average current with RR is lower than those of other approaches, due to less number of sequence transmissions. The performance gaps between all approaches is small because the connection interval settings are the same.

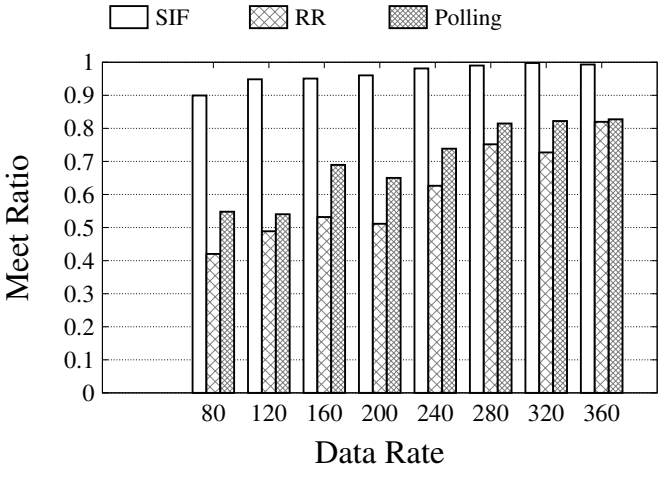
As shown in Fig. 10(b), our proposed SIF achieves the highest meet ratio of all the studied approaches. The meet ratio of SIF increases with the data rate. This occurs because the EIMA method sets a longer connection interval, which causes longer blocking times with low data rates. However, when the data rate is increased, the blocking effect decreases with the connection interval, but the required current increases. The meet ratio of the Polling method is superior to that of the RR method, because the Polling assignment considers the priority of each node in each round, and thus, the highest-priority node can be served more during the hyper-period. The meet ratio of the SIF method is higher than those of the Polling and RR methods, by up to 48% and 35%, respectively.

5.5 Varied Number of Nodes

This section presents a comparison of the proposed SIF scheduler with EIMA of MEI against iOS using a Round

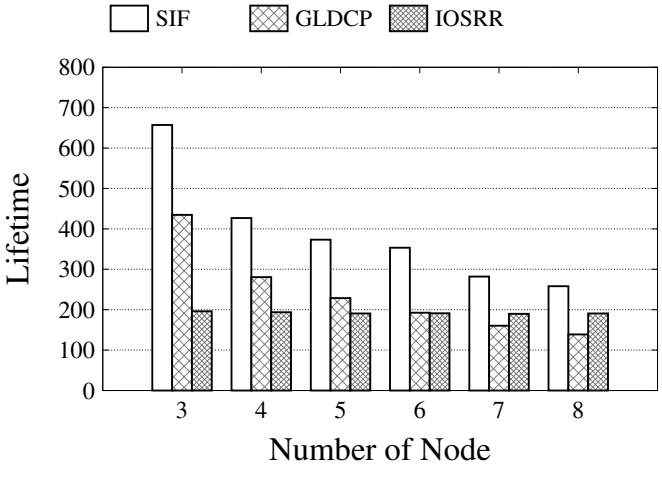


(a) Lifetime

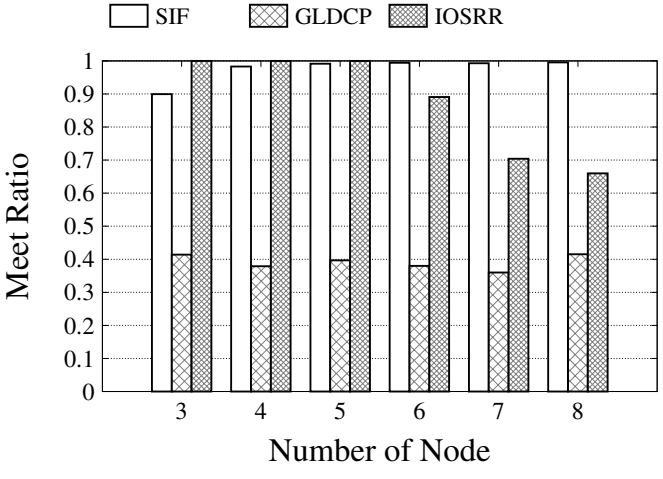


(b) Meet Ratio

Fig. 10. Write-Request Schedule



(a) Lifetime



(b) Meet Ratio

Fig. 11. Multiple Nodes

Robin (IOSRR) scheduler and a greedy approache, which uses Polling with Greedy service interval settings conducted through LDC connection interval scaling (GLDCP) with varying numbers of nodes in the networks. BLE can connect up to eight nodes to request more sensing data. Fig. 11 shows the meet ratio and lifetime of these two approaches with three to eight nodes at a data rate of $80\ bytes/s$.

Fig. 11(a) shows the network lifetimes produced using these approaches; the performance levels of SIF and GLDCP decreased with an increase in the number of nodes. The rationale is that connection intervals decrease with increasing numbers of nodes with both LDC and EIMA. LDC estimates the connection interval as the ratio of a service interval of a node to the number of nodes, and EIMA estimates the connection interval of a node as the ratio of its service interval to that of its weight to the summation of all weights of all nodes. In contrast to GLDCP and SIF, iOSRR produces network lifetime that is stable because of the static connection interval setting. The network lifetimes produced using our proposed method are approximately 51% and 235% times longer than those produced by GLDCP and iOSRR.

Fig. 11(b) shows that the meet ratios produced by SIF increased with the number of nodes. Although the connection interval from GLDCP decreased according to the number of nodes, the delay time of the lower-priority nodes increased with the number of nodes through the Polling scheduler. Thus, the meet ratio of GLDCP was stable with increased nodes, whereas iOSRR set the connection interval as $20\ ms$ without considering the number of nodes, so that the meet ratio decreased with an increasing number of nodes. The meet ratio produced by SIF was superior to that of GLDCP and IOSRR by up to 60% and 33%. This shows that our approach can extend the network lifetime without sacrificing the quality of service, even if many nodes are in the network.

6 Conclusion

This paper proposed an energy-efficient scheduling framework for multiple BLE nodes. With a BLE slave node with multiple applications, each of which has different data rates and latency sensitivities, we presented a demand packet estimation (MEI) to identify the required connection interval. We presented a connection interval assignment (EIMA) method for balancing energy consumption and the blocking time for multiple BLE nodes in a network. To resolve data collisions in the physical channel, we also proposed a non-preemptive write-request scheduler (SIF) that forbids certain events to balance blocking time and response latency. The capabilities of the proposed algorithms were evaluated by conducting several experiments involving synthesized workloads; encouraging results were obtained regarding energy minimization. These experiments showed that the energy saving of our approach prolonged the network lifetime by 170% compared with a default fixed connection interval setting, such as iOS and SPP-over-BLE profile. We will continue to investigate the power management challenges of more complex topologies with multiple protocols, such as mobile cloud computing for the Internet of Things.

References

[1] N. M. Phuong, M. Schappacher, A. Sikora, Z. Ahmad, and A. Muhammad, "Proceedings of real-time water level monitoring using low-power wireless sensor network," in *Proceedings of Embedded World Conference*, 2015.

[2] T. Instruments, *Texas Instruments CC2540/41 BluetoothR Low Energy Software Developers Guide v1.3.2*, 2013.

[3] C. S. Corporation, *Bluetooth Low Energy (BLE) 1.0*, 2014.

[4] Bluegiga, "Spp-over-ble application note@ONLINE," 2013.

[5] K. Cho, G. Park, W. Cho, J. Seo, and K. Han, "Performance analysis of device discovery of bluetooth low energy (ble) networks," *Computer Communications*, 2015.

[6] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream congestion control in wireless sensor networks through cross-layer optimization," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 4, pp. 786–795, 2007.

[7] P. M. Østhus, "Concurrent operation of bluetooth low energy and ant wireless protocols with an embedded controller," Master's thesis, Norwegian University of Science and Technology Department of Electronics and Telecommunications, June 2011.

[8] K. Mikhaylov, "Accelerated connection establishment (ace) mechanism for bluetooth low energy," in *Proceedings of Personal, Indoor, and Mobile Radio Communication (PIMRC), IEEE 25th Annual International Symposium on*. IEEE, 2014, pp. 1264–1268.

[9] B. Jang, J. B. Lim, and M. L. Sichitiu, "An asynchronous scheduled mac protocol for wireless sensor networks," *Computer Networks*, vol. 57, no. 1, pp. 85–98, 2013.

[10] P. Kindt, D. Yunge, M. Gopp, and S. Chakraborty, "Adaptive online power-management for bluetooth low energy," in *Proceedings of Computer Communications (INFOCOM), IEEE Conference on*. IEEE, 2015, pp. 2695–2703.

[11] J. Yang and S. Ulukus, "Optimal packet scheduling in an energy harvesting communication system," *Communications, IEEE Transactions on*, vol. 60, no. 1, pp. 220–230, 2012.

[12] A. El Gamal, C. Nair, B. Prabhakar, E. Uysal-Biyikoglu, and S. Zahedi, "Energy-efficient scheduling of packet transmissions over wireless networks," in *Proceedings of INFOCOM. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2002, pp. 1773–1782.

[13] Y. Wu, R. Kannan, and B. Krishnamachari, "Efficient scheduling for energy-delay tradeoff on a time-slotted channel," Master's thesis, University of Southern California, Los Angeles, California, April 2015.

[14] R. Aquino-Santos, A. G. Potes, V. Rangel-Licea, M. A. García-Ruiz, L. Villaseñor-González, and A. Edwards-Block, "Wireless communication protocol based on edf for wireless body sensor networks," *Journal of applied research and technology*, vol. 6, no. 2, pp. 120–128, 2008.

[15] X. Zhu, S. Han, P.-C. Huang, A. K. Mok, and D. Chen, "Mbstar: A real-time communication protocol for wireless body area networks," in *Proceedings of Real-Time Systems (ECRTS), 23rd Euromicro Conference on*. IEEE, 2011, pp. 57–66.

[16] E. Uysal-Biyikoglu, B. Prabhakar, and A. El Gamal, "Energy-efficient packet transmission over a wireless link," *IEEE/ACM Transactions on Networking (TON)*, vol. 10, no. 4, pp. 487–499, 2002.

[17] B. Zhang, X. Wan, J. Luo, and X. Shen, "A nearly optimal packet scheduling algorithm for input queued switches with deadline guarantees," *Computers, IEEE Transactions on*, vol. 64, no. 6, pp. 1548–1563, 2015.

[18] F. Shan, J. Luo, and X. Shen, "Optimal energy efficient packet scheduling with arbitrary individual deadline guarantee," *Computer Networks*, vol. 75, pp. 351–366, 2014.

[19] M. Collotta, G. Pau, and G. Scatà, "Deadline-aware scheduling perspectives in industrial wireless networks: A comparison between ieee 802.15. 4 and bluetooth," *International Journal of Distributed Sensor Networks*, 2013.

[20] L. K. Tang Hong-wei, Sun Cai-xia and L. Yong-peng, "Mpt-mac: A multiple packets transmission mac protocol for wireless sensor networks," in *Proceedings of SENSORCOMM: The Fifth International Conference on Sensor Technologies and Applications*, 2011, pp. 58–66.

[21] A. Dementyev, S. Hodges, S. Taylor, and J. Smith, "Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario," in *Proceedings of Wireless Symposium (IWS), IEEE International*. IEEE, 2013, pp. 1–4.

[22] D. Giovanelli, B. Milosevic, and E. Farella, "Bluetooth low energy for data streaming: Application-level analysis and recommendation," in *Proceedings of Advances in Sensors and Interfaces (IWASI), 6th IEEE International Workshop on*. IEEE, 2015, pp. 216–221.

[23] S. C. Ergen and P. Varaiya, "Tdma scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[24] Y. Sadi and S. C. Ergen, "Optimal power control, rate adaptation, and scheduling for uwb-based intravehicular wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 62, no. 1, pp. 219–234, 2013.

[25] O. Chipara, C. Lu, and G.-C. Roman, "Real-time query scheduling for wireless sensor networks," *Computers, IEEE Transactions on*, vol. 62, no. 9, pp. 1850–1865, 2013.

[26] M. Perillo and W. B. Heinzelman, "Asp: An adaptive energy-efficient polling algorithm for bluetooth piconets," in *System Sciences. Proceedings of the 36th Annual Hawaii International Conference on*. IEEE, 2003, pp. 10–pp.

[27] M. Doudou, M. Alaei, D. Djenouri, J. M. Barcelo-Ordinas, and N. Badache, "Duo-mac: Energy and time constrained data delivery mac protocol in wireless sensor networks," in *Proceedings of Wireless Communications and Mobile Computing Conference (IWCMC), 9th International*. IEEE, 2013, pp. 424–430.

[28] D. Contreras and M. Castro, "Adaptive polling enhances quality and energy saving for multimedia over bluetooth," *Communications Letters, IEEE*, vol. 15, no. 5, pp. 521–523, 2011.

[29] M. Collotta and G. Pau, "Bluetooth for internet of things: A fuzzy approach to improve power management in smart homes," *Computers & Electrical Engineering*, 2015.

[30] A. K. Jacob and L. Jacob, "Energy efficient mac for qos traffic in wireless body area network," *International Journal of Distributed Sensor Networks*, 2015.

[31] J. L. Sandeep Kamath, *Measuring Bluetooth Low Energy Power Consumption*, TEXAS INSTRUMENTS, 2012, aN092.

[32] C. Gomez, I. Demirkol, and J. Paradells, "Modeling the maximum throughput of bluetooth low energy in an error-prone link," *Communications Letters, IEEE*, vol. 15, no. 11, pp. 1187–1189, 2011.

[33] A. Paul and R. Cyriac, "A review of packet scheduling schemes in wireless sensor networks," *International Journal of Advanced Research in Computer and Communication Engineering*, pp. 5283–5286, 2014.

[34] L. Tang, Q. Guan, S. Jiang, and B. Guo, "A deadline-aware and distance-aware packet scheduling algorithm for wireless multimedia sensor networks," *International Journal of Distributed Sensor Networks*, 2015.

[35] D. Fernandes, A. Ferreira, J. Mendes, and J. Cabral, "A wireless body sensor network based on dynamic power control and opportunistic packet scheduling mechanisms," in *Proceedings of Industrial Technology (ICIT), IEEE International Conference on*. IEEE, 2015, pp. 2160–2165.

[36] F. Hoflinger, G. U. Gamm, J. Albesa, and L. M. Reindl, "Smartphone remote control for home automation applications based on acoustic wake-up receivers," in *Proceedings of Instrumentation and Measurement Technology Conference (I2MTC), IEEE International*. IEEE, 2014, pp. 1580–1583.

Jing-Ho Chen is working on technology industry. He received his M.S. degree in electrical engineering from National Taiwan University of Science and Technology at 2016, and was supervised by Dr. Ya-Shu Chen. He earned his B.S. degree in electrical engineering from National Taiwan University of Science and Technology at 2014. His research interests include wireless sensor networks and scheduling algorithms for embedded systems.

Ya-Shu Chen joined Department of Electrical Engineering, National Taiwan University of Science and Technology, at August 2007. She currently serves as an Associate Professor. Ya-Shu Chen earned her BS degree in computer information and science at National Chiao-Tung University in 2001. Then, she studied in Department of Computer Science and Information Engineering, National Taiwan University, and was supervised by Prof. Tei-Wei Kuo. She successfully defended her master thesis and doctoral dissertation at 2003 and 2007, respectively. Her research interest includes operating systems, embedded storage systems, and hardware/software co-design.