2009 Third International Conference on Sensor Technologies and Applications

An Enhanced Cross-Layer Protocol for Energy Efficiency in Wireless Sensor Networks

Jaehyun Kim
Dept. of Electrical & Electronic Eng.
Yonsei University
Seoul, Korea
Email: macross7@yonsei.ac.kr

Jaiyong Lee
Dept. of Electrical & Electronic Eng.
Yonsei University
Seoul, Korea
Email: jyl@yonsei.ac.kr

Seoggyu Kim
Dept. Information & Computer Eng.
Andong National Univerity
Andong, Koera
Email: sgkion@andong.ac.kr

***Abstract*—In wireless sensor networks, the simplified and energy efficient protocol should be designed in order to maximize the network lifetime because of its stringent resource constraints, ultra power limitation, and tiny embedded devices. In this paper, we propose an enhanced cross-layer protocol for energy efficiency in wireless sensor networks by integrating medium access control and routing protocol. Our proposed protocol utilizes a synchronous medium access control scheme by using the adaptive duty cycling technique to improve energy efficiency and solve long end-to-end delay problem. We also design a tree-based energy aware routing algorithm to prolong the network lifetime in our protocol. Simulation results show that the proposed protocol outperforms other existing algorithms in terms of energy efficiency and latency.**

***Keywords*-wireless sensor network; cross-layer protocol; duty cycle; energy efficiency;**

I. Introduction

In recent years, since the ultra power and resource limitation of wireless sensor networks (WSNs), there has been a great interest in the study of maximizing the lifetime of WSNs and a lot of research efforts have been made to improve energy efficiency in medium access control (MAC) and routing protocols for WSNs. S-MAC [2] and T-MAC [3] have been proposed for WSNs to decrease the wasteful energy expenditure and they share the schedule information that specifies the cycle of active and sleep period, called *duty cycle*, through a SYNC or beacon packet to improve energy efficiency. While its energy efficiency has better performance than typical 802.11 based MAC protocols [1], S-MAC has high delay problem in many-to-one (or few) multi-hop communication pattern of WSNs and it still brings about energy waste due to the fixed duty cycle scheme. T-MAC is a timer based protocol inspired by S-MAC and its duty cycling can allow T-MAC to adjust to fluctuations in the dynamic network traffic. Although, T-MAC shows better results under variable traffic loads, it still undergoes the same delay and energy wastage problems as S-MAC. As in the case of MAC protocols, the several routing protocols, such as Ad hoc On-demand Distance Vector (AODV) [4] and Dynamic Source Routing (DSR) [5], have been developed and suggested for WSNs. These protocols are on-demand routing algorithms in which routes are only discovered when they are actually needed. Thanks to ultra energy limitation and inefficient repairing scheme against data delivery in dynamic network topology changes of WSNs, these routing protocols cannot be appropriately adapted for WSNs. In this paper, we propose an *Enhanced Cross-Layer Protocol, so called ECLP*, for energy efficiency in wireless sensor networks by integrating MAC and routing protocol. For reducing energy wastage due to idle listening and overhearing and for alleviating long delay, our proposed protocol uses an adaptive duty cycle scheme with the adaptive time-out and *RRTS (Reservation Request-to-Send)*. Moreover, a tree-based energy aware routing algorithm is developed in ECLP to maximize the network lifetime but minimize the control overhead required for data delivery.

II. The Proposed Protocol

ECLP is an integrated MAC and routing protocol for energy efficient data delivery to the sink node with adaptive duty cycling and tree-based energy aware routing algorithm while minimizing overhead cost and latency. The basic MAC operation of ECLP is based on AD-MAC [7][8] in which we have previously proposed. In this paper, we have extended AD-MAC to enhance the performance of MAC by adopting the adaptive time-out scheme considering both energy efficiency and latency. The operation of ECLP protocol will be particularly described in the following subsections.

A. Network Model

A wireless sensor network (WSN) is modeled as an directed graph $G(V, E)$, where $V = S \cup N$ is the number of sensor nodes in the network, *S* is the number of sink nodes, *N* is the number of sensor nodes, and *E* is the number of links in the network. There can be multiple sink nodes in WSNs but each sensor node is assigned to only one sink node since asymmetric communication paradigm generally occurs in many nodes-to-one sink communication pattern. We assume that the network comprises randomly distributed nodes with a single sink node and data aggregation is not considered

978-0-7695-3669-9/09 $25.00 © 2009 IEEE
DOI 10.1109/SENSORCOMM.2009.106

here. We also consider that all the sensor nodes have the same transmission range and initial power capability.

B. Tree-based Routing Algorithm

In ECLP, a tree-based energy aware routing algorithm is executed. It is formed by the *SYNC* and $SYNC_{reply}$ packet. In synchronous MAC protocols, such as S-MAC and T-MAC, the SYNC packet is used only for synchronization. However, the *SYNC* of ECLP is applied to not only routing configuration and management but also synchronization. The $SYNC_{reply}$ of ECLP is exploited to confirm the tree path configuration and recognize the total number of hops from the sink node on each branch node of the tree network. In the following subsections, the rouging algorithm of ECLP is presented in more detail.

1) Determining the Next Forwarding Node: To select the next forwarding node (parent node) for tree-based energy aware routing path to the sink node, ECLP uses the routing cost supplied with information such as transmission cost *(link cost)* and energy cost *(node cost)* together. A routing algorithm using only hop count cannot optimize the energy consumption across the network, especially in non-uniform traffic conditions. Thus, maximizing the network lifetime is the primary goal of our algorithm and we adopt the routing cost function, r_cost with r_lth. In the case of WSNs, the resultant path with many short-range links may perform worse than a path with fewer long-range links in terms of latency as well as energy consumption. This is because the path with many short-range links would cause more link errors that result in more retransmissions [6]. In this case, the link layer retransmissions on a specific link essentially ensure that the transmission energy spent on the other links in the path is independent of the error rate of that link. In ECLP, the link error rate parameter is employed as the *link cost* and the energy cost parameter provided by the packet receiving/transmitting energy and residual energy is exploited as the *node cost* together. Since the number of transmission on each link is independent of the other links and is geometrically distributed, the routing cost of node *i*, r_cost_i is defined in ECLP as follows:

$$r\_cost_i = \frac{E_{T\_i}}{{E_{r\_i}}^{\alpha}(1 - p_{err\_i})^{\beta}} \quad (1)$$

where, E_{T_i} is the unit packet transmission cost of node *i* (e.g., E_T = E_{recv} + E_{trans}, E_{recv} is the consumed energy for packet reception and E_{trans} is the consumed energy for packet transmission), E_{r_i} is the residual energy of node *i*, and p_{err_i} is the packet error probability of the link between node *i* and *j*. Also, α and β are nonnegative weighting factors for setting the relative between *link cost* and *node cost* in the range [0, 1]. Calculating and choosing this minimum routing cost function (1) is equivalent to choosing a minimum cost path from node *i* to the sink node. This routing cost function is fully localized, distributed and computed with only its neighbor nodes from a node.

In ECLP, there are three steps for choosing the next relay node (parent node) for routing destined for the sink node. First, a node checks its residual energy. If the residual energy of the node is less than the energy threshold of its *Danger state* (Th_D), which implies that the node has no more energy to take more transmission jobs with other nodes, the node simply discards the received request. Secondly, the node received *SYNCs* from its neighbor nodes checks the r_lth in the *SYNCs*. Then, it chooses one neighbor node having the smallest value of r_lth in the *SYNC* as the next relay node. Finally, if there are the multiple nodes with the same r_lth value, it next compares the r_cost in the *SYNCs* among the multiple nodes. Then, it selects one neighbor node with the minimum value of r_cost as the next relay node. After the next forwarding node (parent node) is determined, it adds the chosen relay node to its neighbor management table with the r_lth value. The smallest value of r_lth means the minimum hop count needed to reach the sink node. The minimum value of r_cost implies the minimum energy cost for increase energy efficiency. Routing algorithm based on global information may provide the optimized path from the source to destination pair, but it can result in long configuration latency and high overhead cost for routing management. Based on this local information, ECLP may not provide the optimal path but it can select the next forwarding node (parent node) through which the overall energy cost destined for sink node is minimized. Therefore, ECLP can prolong the network lifetime and reduce the overhead cost.

2) Path Set-up Phase: In ECLP, the *SYNC* and $SYNC_{reply}$ packet are used for setting-up the tree-based energy aware routing path in the network. The *SYNC* of ECLP has five new fields compared with the SYNC in S-MAC: $\{r\_lth, r\_cost, thresholds, parent, status\}$, indicating the sensor node's routing length, its routing cost, its energy thresholds, its parent node in the branch tree, and its status. The $SYNC_{reply}$ of ECLP is exploited to confirm the tree path configuration and recognize the total number of hops (hop count) from the sink node on each branch node of the tree network. The $SYNC_{reply}$ of ECLP is similar to the SYNC of ECLP but it has two different fields from the *SYNC* of ECLP: $\{r\_lth_{total}, r\_cost, thresholds, sink, status\}$, indicating the total number of the branch tree's routing length (hop count of the leaf node), its routing cost, its energy thresholds, its root node (sink node) in the branch tree, and its status. The length of all new fields is one byte respectively, except the length of the *status* field is two bits. Thus, the *status* field shows four states, {*intermediate node*, *leaf node*, *Danger state*, *Emergency state*}. *Intermediate node* and *leaf node* indicate that the node plays the part of the intermediate node and the leaf node on the branch tree. *Danger state* means the residual energy of a node is less than the energy threshold of its *Danger state* (Th_D). So, the node in *Danger state* does not participate in data

transmission jobs with its neighbor nodes. It just sends its own sensed data to the sink node. *Emergency state* signifies the broken node has occurred and the routing path has lost. Thus, the local path recovery phase is executed in this case. The local path recovery phase will be described in the following subsection. The *thresholds* field is comprised of two values of the energy thresholds, Th_R and Th_D. Th_R is used for the adaptive time-out mechanism. When the residual energy of a node is larger than the energy threshold of its *Relief state* (Th_R) in the *SYNC* packet, the node sets its adaptive timer (T_A) to T_{min} for prolong energy efficiency. The more detail description of this function will be explained in Section II.C. Th_D is used for the participation of a node's data transmission and local path recovery phase. Th_R and Th_D are pre-determined in the *SYNC* and they are dependent on the applications of WSNs. (e.g., Th_R = 50% of the initial node energy, Th_D = 15% of the initial node energy)

Initially, all nodes are randomly deployed and have both the r_lth of 255 and the r_cost of 255 in the *SYNC* while the sink node has the r_lth of 0 and the r_cost of 255 in the *SYNC*. Actually, r_lth and r_cost are one byte, respectively in the *SYNC* so that they have the maximum value of 255, respectively. In the first step, the routing path configuration is developed between the sink node and its neighbor nodes. The sink node periodically broadcasts the *SYNC* to its neighbor nodes with the initial r_lth and r_cost value. When the neighbor nodes receive the *SYNC* from the sink node, they first add the sink node to their neighbor management table with the r_lth of 0 as their parent node. Secondly, they calculate their own r_cost values and store them in the *SYNCs*. Afterward, they increment the r_lth values in the *SYNCs* by one to update. Then, they broadcast the *SYNCs* to their neighbor nodes. In the second step, the routing path configuration is developed between the nodes with the r_lth of 1 and their neighbor nodes. The node received the *SYNCs* rom its neighbor nodes first checks the r_lth values in the *SYNCs* and selects the node with the r_lth value of 1 as its parent node. If there are multiple nodes with the r_lth of 1, the node next checks the r_cost value in the *SYNCs* among them. Then, it selects one node with the minimum value of r_cost as its parent node. After its parent node is found, it adds its parent node to its neighbor management table with the r_lth of 1 for primary routing. Moreover, it adds the nodes which has the same r_lth of 1 but has the different (larger) value of r_cost or which has the different (larger) r_lth value to its neighbor management table with r_lth of 1 for alternative routing. Any node acts on only the first *SYNC* with the same ID and ignores any subsequent *SYNCs*.

In addition, the node calculates its own r_cost values and stores its own r_cost in the *SYNCs*. Afterward, it increments the r_lth value in the *SYNC* by one (e.g., the r_lth value of 2) to update and broadcasts the SYNC to its neighbor nodes like the first step. This r_lth update process is repeated and the local routing information with the parent-to-child node pair (to the sink node) is made in the neighbor management table. In the final step, a leaf node sends the $SYNC_{reply}$ to the sink node through each node on the branch tree in order to confirm the tree path configuration and recognize the total number of hops (hop count) from the sink node on each branch tree. The more detail description of this function will be presented in Section II.C.

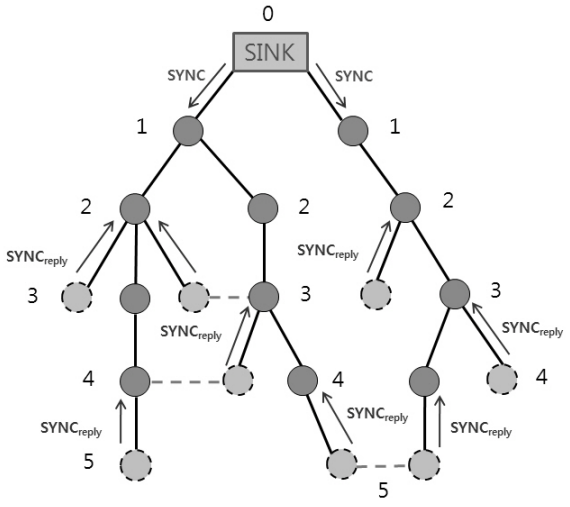

Figure 1. Tree-based energy aware routing algorithm in ECLP

Consequently, the tree-based energy aware routing path configuration is completely established from the sink node to leaf nodes across the whole network. In the final step, Note that only local information and neighbor management table is utilized in ECLP and the global tree structure is not maintained by any node. Fig. 1 shows one example of initial path configuration in ECLP. The tree-based path configuration process with the *SYNC* and $SYNC_{reply}$ is depicted in Fig. 1. In Fig. 1, the solid lines mean the primary routing path and the dashed lines signify the alternative routing path. Also, the solid-circles mean intermediate nodes and the dashed circles imply leaf nodes in the branch tree. Note that during the tree-based energy aware routing path construction. ECLP only requires each node to broadcast once to avoid wastefully overlapping *SYNCs*.

3) Data Forwarding Phase: ECLP is a kind of table-driven approach and the routing table is established in the path set-up phase as the neighbor management table. In the neighbor management table, only local information from *SYNC* packets between a node and its neighbor nodes is utilized. The tree-based routing table indicating the parent and child node pair information is exploited for data delivery from a sensor node to the sink node. Once the routing tree has been built, a node which has data to the sink node just sends them to its parent node and the data are finally delivered to the sink node through the constructed routing tree. If the residual energy of the node is less than the

energy threshold of its *Danger state* (Th_D), which implies that the node has no more energy to take part in for more transmission jobs with other nodes, the node simply discards the received request. Then, the node changes its state to *Danger state* of the status field in the *SYNC* and it sends the $SYNC_{reply}$ packet to the sink node in order to indicate it becomes the leaf node on the branch tree from the sink node. In other words, the node (leaf node) only sends its own sensed data to the sink node.

4) Path Recovery Phase: Due to a node's mobility or breakdown, when a link on the active branch route is broken, the node tries to locally repair the broken link. When a child node starts to send data to its parent node, it first sends a RTS to its parent node. If the child does not receive a CTS from its parent node for sending a RTS three times, the child node can be aware that its parent node has been lost. Then, the child node begins the local path recovery process. First of all, if the child node has another parent node destined for the sink node in its neighbor management table (as alternative routing path), it simply changes the new parent node and tries to send data to the new parent node. At this moment, the r_lth value of the broken node is edited to 255 (namely, unreachable) in the neighbor management table. In this case, there is no additional signal overhead and fast path recovery is possible.

However, if there are no alternative parent nodes in the neighbor management table, another path recovery mechanism is performed. A *PERR (Path ERRor)* control packet is used for the path recovery in ECLP. The *PERR* is almost same as a *SYNC* of ECLP and it indicates the *PERR'* sender lost the link. The difference between the *PERR* packet and *SYNC* packet is to point out the lost node's ID in the field of the control packet. If the broken node is only one parent node destined for the sink node, the child node changes its state to *Emergency state* of the *status* field in the *PERR* indicating the broken node's ID and broadcasts the *PERR* packet. In *Emergency state*, the nodes cannot send any data and they have the r_lth of 255. In this case, only *SYNCs* from other neighbor nodes can release this *Emergency state* since the *SYNCs* from others have reliable the tree routing information with r_lth, r_cost and *parent ID*. Then, the nodes received *PERRs* compare their r_lth values with the r_lth of *PERRs* and check their residual energy. If the r_lth value of the *PERR* is larger and their residual energy is larger than the energy threshold of its *Danger state* (Th_D), the *PERR* is ignored and the node having the larger r_lth broadcasts the *SYNC* with its tree routing table. The node having the larger r_lth implies that it has the alternative routing path (parent node) destined for the sink node. Therefore, the new tree routing path is found and the broken path is recovered in the end. Otherwise, in other words, if the r_lth value of the *PERR* is smaller or the node's residual energy is less than the energy threshold of its *Danger state* (Th_D), it means that the nodes received the *PERR* have lost the routing path to the sink node likewise. In the same manner, they edit the r_lth value of the broken node to 255 in their neighbor management table and they can recognize which node has gone from the broken node's ID designated in the *PERR*. Then, they also try to find the alternative parent node destined for the sink node repeatedly. If it succeeds, they broadcast the *SYNC* and the lost path finally has been recovered. However, if they cannot find the alternative parent node, they go to the *Emergency state* with the r_lth of 255 and broadcast *PERRs* to their neighbor nodes again until they obtain the alternative parent node in a certain period. If they fail to repair the lost link in the certain period, they stop the path recovery scheme and wait the initial path configuration process by the sink node. During this situation, any node in the *Emergency state* stops data delivery and store them. When the node obtained the new alternative parent node to the sink node, it directly broadcasts a *SYNC* to update the new path configuration. After all nodes in the *Emergency state* receive the *SYNC*, they change their old paths into the new detour path to reach the sink node and the tree routing is finally reconfigured in the tree network. The basic local path recovery algorithm is presented in Fig. 2.

```
while (data forwarding is needed) do
1       if (node i finds an upper node's link lost)
2          if (node i has another (alternative) upper
              node || receives a new SYNC)
              /* Problem Solved */
3             turns off Emergency state;
4             sends a SYNC to its neighbor nodes;
           else
           /* Repair the lost link */
5             sends a PERR to its neighbor nodes;
6             turns on Emergency state;
7             waits a SYNC from its neighbor nodes;
8       if (node j receives a PERR)
9          if (r_lth of itself < r_lth of PERR &&
              the residual energy of node j is larger
              than the energy threshold of node j's
              Danger state (Th_D)
              /* Problem Solved */
              Step 3;
           else
              /* Repair the lost link */
              Step 5;
done
```

Figure 2. Local path recovery algorithm

C. Adaptive Synchronous MAC Scheme

ECLP performs the adaptive synchronous MAC scheme. In ECLP, the active period ends when no activation event has occurred for a time-out period (T_A). In other hands, if there are no data to send or receive until the timer's expiration, the node goes to sleep to reduce the unnecessary time and energy wasted in idle listening. The MAC operation of ECLP is based on AD-MAC (Adaptive Duty cycling

synchronous MAC) [7][8] and it is enhanced by using *RRTS (Reservation Ready-to-Send)* and the adaptive time-out mechanism in ECLP. ECLP allows non-intended recipients to avoid overhearing by returning to sleep immediately when they catch the control packets such as a RTS, CTS, and *RRTS*. In unidirectional multi-hop communication pattern from numerous nodes to the sink, the per-hop latency can be increased. This long latency problem can be solved by using a *RRTS* packet with the adaptive timer (T_A) to adapt traffic conditions in ECLP. The *RRTS* packet is similar to the FRTS (Future Request to Send) packet in T-MAC [3] but there are two differences. First, the *RRTS* packet lets other nodes know there are remaining data to deliver. If a node overhears a CTS packet destined to another node, it immediately sends a *RRTS* packet to the next forwarding node. The *RRTS* packet contains the information about the duration of data communication so that the node received the *RRTS* packet can go to sleep until the previous data transmission is completed. Then, it wakes up and receives data from the backward node (child node). In T-MAC, if a node overhears a CTS packet destined to another node, it sends a FRTS packet to the next forwarding node. A node received the FRTS packet should be awake by the next time to communicate with the prior node. It results in wasteful energy consumption. Secondly, the *RRTS* packet includes the value of a new time-out so that this new value can be adapted to the next forwarding nodes to change a time-out threshold (T_A) for reducing delay for the next data communication. This technique can adapt to traffic fluctuations and have good influence on dealing with traffic condition for shortening latency.

To solve the early sleeping problem which was explained in T-MAC, the interval *TA* must be long enough to receive at least the start of the CTS packet and the decision of *TA* is presented as follows [3]:

$$TA > Ct + R + T \tag{2}$$

where, *Ct* is the contention interval, *R* is the length of a RTS packet, *T* is the very short guard time. In ECLP, we adopt the adaptive time-out scheme based on r_lth that implies the hop count (distance) from the sink node. In short, the adaptive timer, T_A is employed in proportion to r_lth in ECLP, namely, the smaller the r_lth, the smaller value of the timer (T_A). It means that the nodes closer to the sink node have the larger values of the timer (T_A) and they have longer waiting time for data delivery. This scheme can decrease the end-to-end delay but spend more energy. So, this algorithm is only performed when the residual energy of a node *i* is larger than the energy threshold of its *Relief state* (Th_R) in the *SYNC* packet. Initially, the sink node broadcasts a *SYNC* packet to nodes in the network and the tree-routing path is built up. Nodes can obtain the distance (hop count) from the sink node from the r_lth value in the *SYNC*. However, they cannot know the total hop count of the branch tree. If the

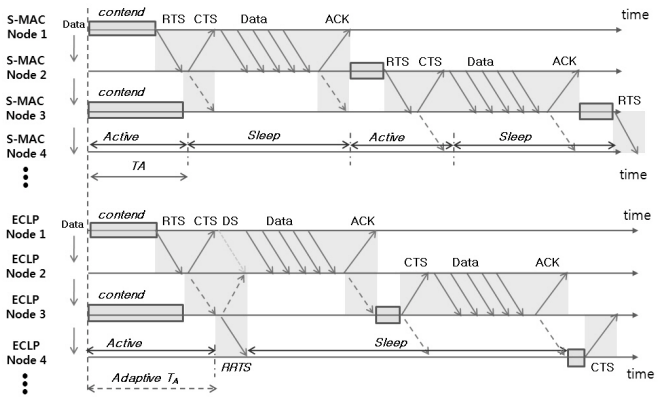

Figure 3. Adpative duty cycle scheme in ECLP

total hop count from the sink node and a node's hop count information on the branch tree is recognized, the node can set up the adaptive timer (T_A) in proportion to the hop count. The procedure of acquiring the total hop count value is the following. After receiving the *SYNC* for the first time, a node *i* sets up its timer to T_i. T_i counts down when the channel is idle. When the timer T_i times out, the node *i* increases its r_lth (hop count) by one and broadcasts the *SYNC*. If a node *k* receives a *SYNC* from node *i* indicating that its parent node is node *k*, then *k* marks itself as a intermediate node. Otherwise, the node marks itself as a leaf node. This process continues until each node broadcasts once. If a node becomes a leaf node, it signifies its r_lth is the total hop count from the sink node in the branch tree. Then, a leaf node sends the $SYNC_{reply}$ to the sink node through each node on the branch tree to let them know the total hop count (r_lth_{total}) of the branch tree. Finally, each node from the sink node and leaf nodes can perceive the total hop count (r_lth_{total}) from the sink node and their hop count. Afterward, each node on the branch tree sets up the adaptive timer, (T_A) in proportion to this information. The adaptive timer of node *i*, (T_{A_i}) is expressed in ECLP as follows:

$$T_{A\_i} = \begin{cases} T_{min} + \gamma(T_{max} - T_{min}) \times (1 - \frac{r\_lth_i + 1}{r\_lth_{total}}), \\ \qquad\qquad i \le r\_lth_{total} - 2 \\ T_{min}, \qquad\qquad \text{otherwise} \end{cases} \tag{3}$$

where, T_{min} = Ct + R + T, T_{max} = $Duty\ cycle$ in the *SYNC* packet, γ is the proportional constant in the range [0, 1] to alleviate energy consumption. In general, the leaf node and its parent node do not need to set up the adaptive timer because these two nodes are the start point in the multi-hop data communication from the leaf node to the sink node. So, the leaf node and its parent node just set up their timer to T_{min}. γ is used for reduce the impact of the adaptive time-out mechanism for expand energy efficiency. Note that this

algorithm is only accomplished when the residual energy of a node *i* is larger than the energy threshold of its *Relief state* (Th_R) in the *SYNC* packet. It results in minimizing energy wastage of the adaptive timer. As we have discussed, the relationship between energy efficiency and latency is trade-off so it depends on the applications or requirement of WSNs. However, this adaptive time-out mechanism with the energy threshold (Th_R) and γ can improve the MAC performance of ECLP in terms of both energy efficiency and latency. Furthermore, if a node overhears a CTS packet from its child node (backward node) or receives a *RRTS* packet from its child node, the node goes to sleep until the completion of the previous data transmission and then it wakes up and directly sends CTS to receive data from its child node. In ECLP, the tree routing path has already been set up so that this mechanism can be executed and useful to reduce the delay of data transmission as well as the control overhead cost.

Fig. 3 briefly shows the adaptive synchronous MAC operation of ECLP compared with S-MAC. Assuming that node 1 sends data to route node 4 through node 2 and node 3, in other words, the tree-based routing path has already been constructed and data are destined for the sink in the end, node 1 operates as typical contention-based MAC protocols. When node 3 overhears a CTS packet from node 2, it knows the next forwarding node (parent node) itself so that it can go to sleep until finishing the previous data transmission. Afterward, node 3 directly initiates data transferring unlike typical contention-based MAC approaches. That is, node 3 directly sends a CTS packet to node 2 after overhearing an ACK packet from the previous node, node 2 and then receives data from node 2. This mechanism reduces the overhead cost of control packets and also improves energy efficiency compared to S-MAC and T-MAC. On the other hand, when node 3 overhears a CTS packet, it immediately sends a *RRTS* packet and goes to sleep until the previous data communication completion. Node 4 can recognize there are data destined for itself and the duration of the communication so that it can go to sleep until completing the previous data transmission. Note that when node 4 wakes up, it directly sends a CTS packet like the previous procedure. Consequently, this mechanism enhances energy efficiency and also decreases the per-hop latency and the control overhead cost as well.

III. Performance Evaluation

The ECLP was implemented in ns-2 [9] to validate and evaluate the performance of the proposed protocol in comparison with IEEE 802.11 MAC and AODV/DSR routing protocols pair and S-MAC and AODV/DSR protocols pair. The simulation was carried out with 20 or 50 nodes and randomly chosen one node moved away in every experiment. This method was considered in order to evaluate mobility effect. In our evaluation, the unicast traffic and asymmetric communication is considered, like in many nodes-to-one sink traffic pattern. Also, energy consumption by the radio turn on/off is mainly focused and not considered by processing or sensing data in our experiment. The average energy consumption is the sum of the average energy consumption in each state: transmitting, receiving, idle listening and sleep respectively. In transmitting and receiving state, data packets and control packets of MAC and routing are included. The basic simulation parameters are shown in Table I.

Table I
Simulation Parameters

Parameter	Value
Number of nodes	20 or 50 nodes
Data transmission speed	20 kbps
Transport layer	UDP
CBR (Constant bit rate)	2 $\sim$ 0.02 packet/sec
PHY (Physical layer)	IEEE 802.11
Antenna	Unit antenna range
Energy consumption in transmitting	36 mW
Energy consumption in receiving	14 mW
Energy consumption in active state	14 mW
Energy consumption in sleep state	0.15 μW
SYNC packet cycle	20 frames
Duty cycle in S-MAC/ECLP	50%
Link error rate	0 $\sim$ 0.25

A. Energy Consumption

In ECLP, the average energy consumption is the sum of data transmitting, receiving, idle listening and sleep state per each node. Therefore, the average energy consumption of IEEE 802.11 based protocols is

$$E_{802\_11} = E_{trans} + E_{recv} + E_{idle} \quad (4)$$

where, E_{trans} is the average energy spent for transmitting data and E_{recv} is the average energy spent for receiving data. Also, E_{idle} is the average energy spent for idle listening and E_{sleep} is the average energy spent for sleep period. In S-MAC and ECLP, duty cycling technique is utilized for reducing unnecessary energy wastage caused by idle listening. The average energy consumption of S-MAC and ECLP is

$$E_{SMAC/ECLP} = E_{trans} + E_{recv} + E_{idle} + E_{sleep} \quad (5)$$

Fig. 4 and Fig. 5 show the average energy consumption with different number of nodes and varying packet rates. IEEE 802.11 based MAC protocols typically waste unnecessary energy because of idle listening. This approach always turns on its radio to wait data transmission and reception. In addition, with low mobility and node density, the energy consumption of DSR and AODV is similar. On the other hand, S-MAC uses the duty cycling technique to reduce energy wastage caused by idle listening so that S-MAC based protocols (DSR and AODV) have better energy efficiency than IEEE 802.11 based approaches. In ECLP, however, the adaptive synchronous MAC scheme effectively executes the

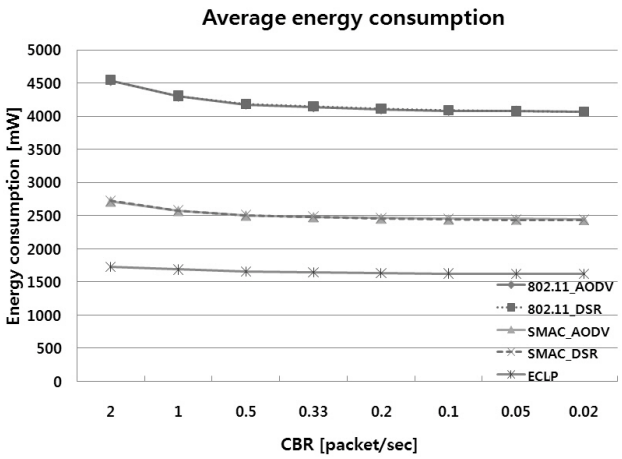

Figure 4. Average energy consumption with 20 nodes

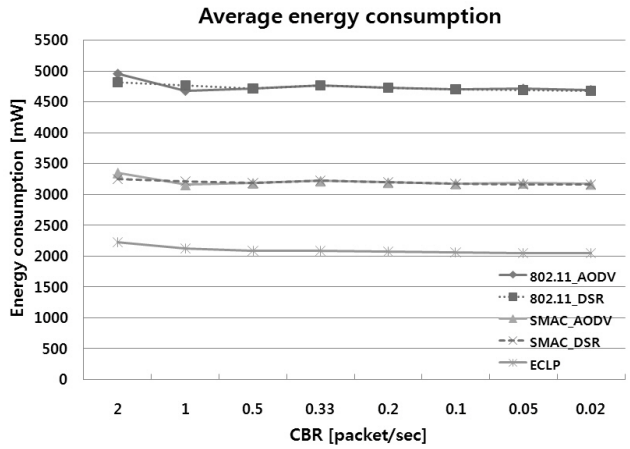

Figure 5. Average energy consumption with 50 nodes

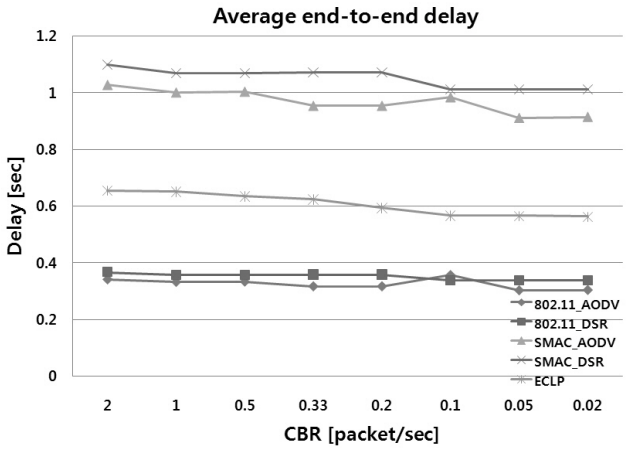

Figure 6. Average end-to-end delay with 20 nodes

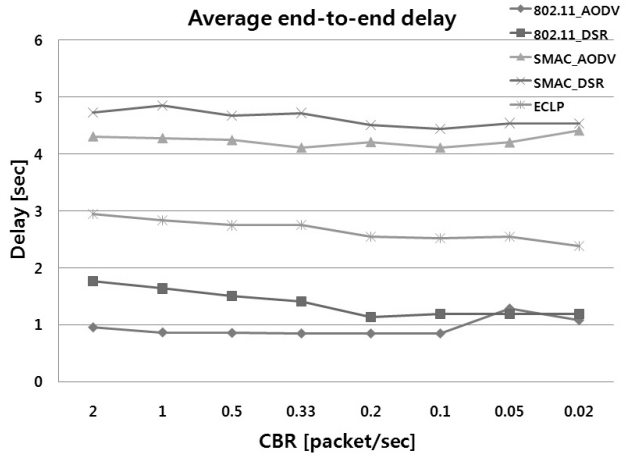

Figure 7. Average end-to-end delay with 50 nodes

adaptive duty cycling by using the *RRTS* and adaptive time-out mechanism with energy thresholds. ECLP also decreases the control overhead cost and idle listening. Moreover, it uses only local *SYNC* packets for tree-based energy aware routing path configuration and management by exploiting r_lth, r_cost and $thresholds$ metrics. Consequently, ECLP outperforms much better than both S-MAC based protocols and IEEE 802.11 based protocols.

B. End-to-End Delay

Fig. 6 and Fig. 7 show the average end-to-end delay with different number of nodes and varying packet rates. In our simulation, AODV slightly has less delay and provides better performance than DSR. However, the relative performance of both protocols with respect to delays is very similar. In general, the duty cycling schemes can improve energy efficiency but they suffer from long end-to-end delay because of the periodical active and sleep state's repetition resulted in the per-hop long delay. Therefore, there is the trade-off relationship between energy efficiency and latency in IEEE 802.11 based approaches and duty cycling mechanisms. The results reveal that S-MAC based protocols typically underperform compared with IEEE 802.11 based protocols. The long end-to-end delay has occurred in S-MAC based approaches because of their duty cycles. However, ECLP has more significant improvements since the adaptive duty cycling scheme utilizes the enhanced *RRTS* technique with the adaptive time-out scheme, which can avoid overhearing and reduce the long end-to-end delay compared with S-MAC based protocols. In addition, due to nodes' mobility or failure, ECLP achieves the fast and effective local path recovery process so that it alleviates the long delay problem in the multi-hop wireless network.

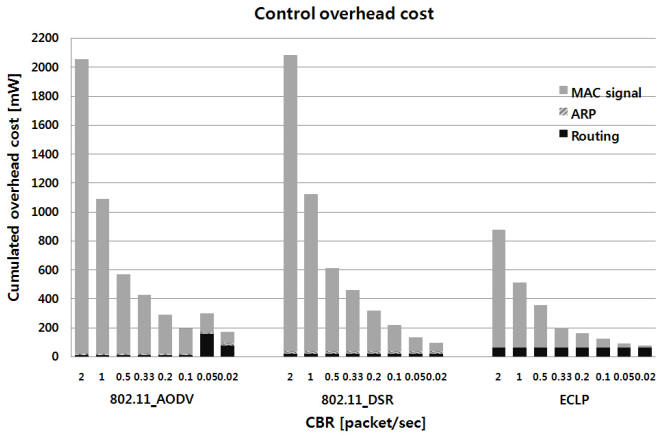

Figure 8. Control overhead cost

C. Control Overhead Cost

The Control overhead cost is defined as the energy spent for control packets from MAC and routing protocol which are MAC signals (*SYNC*, $SYNC_{reply}$, *RTS*, *CTS*, *ACK*, *RRTS*) and routing signals (*RREQ, RREP, RERR, PERR*), except data. The cumulated control overhead cost is shown in Fig. 8. In Fig. 8, the simulation is performed with randomly deployed 20 nodes. The green bar means MAC signaling overhead cost, the red one depicts ARP signaling overhead cost, and the blue one describes routing overhead cost. The control overhead of MAC generally occupies much larger than other parts. The averaged results indicate that the control overhead in IEEE 802.11 based AODV/DSR protocols is much larger than that of ECLP. The reason why ECLP has much smaller the control overhead cost than that of other schemes is that first, ECLP is the integrated protocol by combining MAC and routing protocol. Secondly, it utilizes the effective control packet scheme (*RRTS* and *Direct CTS* technique). Thirdly, only local information is used for tree routing configuration and management and thus ECLP reduces the overhead cost. The control overhead cost of S-MAC based protocols may be smaller than 802.11 based protocols because of massage passing scheme and overhearing avoidance but S-MAC basically has the same mechanism, RTS-CTS-ACK with 802.11 MAC. Therefore, ECLP definitely has lower control overhead cost than S-MAC based schemes and IEEE 802.11 MAC based schemes.

IV. Conclusion and Future Work

In this paper, we propose an enhanced cross-layer protocol for energy efficiency in wireless sensor networks. Both energy efficiency and latency are considered for efficient data delivery in our algorithm. In our protocol, the advanced adaptive duty cycling technique with the adaptive time-out and thresholds reduces long delay and ameliorates energy efficiency. Furthermore, the proposed tree-based energy aware routing algorithm can minimize overhead cost and lengthen the network lifetime. Simulation results have shown that the proposed ECLP has more significant improvement than other schemes, such as IEEE 802.11 and SMAC with AODV and DSR in terms of energy consumption, the end-to-end delay, and the control overhead cost. The future work involves a more detailed analysis with traffic conditions and extended simulation with other parameters, such as success ratio, various mobility, link error rate, and network density. Also, we are implementing our algorithm to practical tiny-OS based sensor nodes to evaluate the performance in detail under the real environment.

Acknowledgment

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2009-C1090-0902-0038).

References

[1] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11-1999 edition.

[2] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Network," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493-506, Jun. 2004.

[3] T. V. Dam and K. Langendoen, "An Adaptive Energy Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. ACM Conference on Embedded Networked Sensor Systems*, Los Angeles, USA, Nov. 2003, pp. 171-180.

[4] C. Perkins, E. B. Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Mobile Ad-hoc Networks (MANET) Working Group, IETF RFC 3561, 2003.

[5] D. Johnson, Y. Hu, and D. Maltx, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," Mobile Ad-hoc Networks (MANET) Working Group, IETF RFC 4728, 2007.

[6] S. Banerjee and A. Misra, "Minimum Energy Paths for Reliable Communication in Muiti-hop Wireless Networks," in *Proc. ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Lausanne, Switzerland, Jun. 2002.

[7] J. H. Kim, J. S. On, S. G. Kim, and J. Y. Lee, "Performance of Evaluation Synchronous and Asynchronous MAC Protocols for Wireless Sensor Networks," in *Proc. International Conference on Sensor Technologies and Applications*, Cap Esterel, France, Aug. 2008.

[8] J. H. Kim, J. Y. Lee, and S. G. Kim, "A Cross-Layer Approach for Efficient Delivery in Wireless Sensor Networks," in *Proc. UKC 2008*, San Diego CA, USA, Aug. 2008.

[9] Ucb/lbnl/vint network simulator—ns (version 2). [Online]. Available: http://www.isi.edu/nsnam/ns/ 03.25.2009