
Wooguil Pak
School of Electrical Engineering & Computer Science, INMC
Seoul National University
Seoul, Korea
wgpak@netlab.snu.ac.kr

Kyong-Tak Cho
School of Electrical Engineering & Computer Science, INMC
Seoul National University
Seoul, Korea
ktcho@netlab.snu.ac.kr

Saewoong Bahk
School of Electrical Engineering & Computer Science, INMC
Seoul National University
Seoul, Korea
sbahk@netlab.snu.ac.kr

Abstract—In this paper, we propose a new centralized routing protocol named WRP that aims at maximizing the network lifetime. In WRP, the sink node is assumed to be with more capabilities and basically collects each node’s estimated wakeup time. This enables the sink node to estimate the wakeup times of all sensor nodes even if some of them are multi-hops away from the sink node. The estimated information is used to find a new route without flooding when some links are broken. Owing to these features, WRP solves the problem of high energy consumption mainly caused by the clock drift in ultra low duty cycled environments. It achieves longer network lifetime independently of the node density because the amount of control traffic does not increase with the node density. We investigate the performance of WRP through extensive simulations and show that WRP increases the network lifetime by more than 10 times against existing routing protocols. WRP can be a very promising routing protocol applicable to the monitoring case like AMR (Automatic Metering Reading) and AMI (Advanced Metering Infrastructure).

I. INTRODUCTION

Wireless sensor networks (WSNs) are currently used in various fields and the automatic metering reading (AMR) is a well-known application [4]. For energy limited applications, many WSN protocols have been proposed to maximize network lifetime. A simple way to increase the network lifetime is to lower the duty cycle when the delay is tolerable. However, it is not possible to guarantee that existing routing protocols can run at their best because most of them are designed without considering ultra low duty cycled environments.

To achieve longer network lifetime in low to ultra low duty cycled networks, we propose WRP (Wide range of duty cycles supporting Routing Protocol). It is a centralized routing protocol that minimizes the control message overhead and consumes very low energy for routing recovery since it efficiently handles the synchronization overhead incurred by the clock drift [3]. Generally, a centralized routing protocol has worse robustness than a distributed routing protocol. Considering that the failure of a sink node corresponds to the whole network failure regardless of what routing protocol was used, we can’t say the centralized routing protocol is inferior to the distributed one any more for WSNs. Rather, the centralized routing protocol can be more suitable for WSNs because a sink node should be with more resources than ordinary nodes such as computational power, memory, and mains power.

WRP adopts the receiver’s wakeup time estimation algorithm that was proposed for W-MAC protocol in our previous work [1]. It extends the estimation algorithm that works between two one-hop (i.e., directly linked) neighbors to cover the multi-hop communication between a sensor node and a sink node. This cross-layered approach enables us to improve the network lifetime and routing maintenance cost.

This paper is organized as follows. We present the issues of a ultra low duty cycled WSN in Section II. In Section III, we show the details of WRP protocol operations and its simulation result in Section IV. Section V concludes our paper.

II. ISSUES FOR ULTRA LOW DUTY CYCLE

The ultra low duty cycle is necessary for a long lifetime of WSN but there are many issues which should be solved before it is used in real network. In this section, we show the issues and characteristics of the ultra low duty cycled WSN, which need to design routing protocol for the application requires a long network lifetime with very low traffic rate.

A. Packet flooding problem

Most of existing routing protocols rely on packet flooding for routing setup or routing recovery1. Because flooding causes a high traffic rate in a short time, it needs high data throughput. However, the throughput of ultra low duty cycled network becomes very low due to very short wakeup time. This results in high collision probability of flooded packets and the long packet forwarding delay is also an obstacle of flooding.

B. High energy consumption by control message

Routing protocols need various control messages to collect the network information, to setup routing path, or to check aliveness of a neighbor node. For ultra low duty cycled network, traffic rate and data throughput are very low. As a result, many control message transmissions cause low data throughput and high energy consumption.

1For proactive routing protocols, they exchange the link state periodically with all neighbor nodes and these overheads can be similar with that of packet flooding. So, we only consider routing protocols that use packet flooding for routing recovery.
III. PROTOCOL DESCRIPTION

WRP can run with from a low duty cycle to a ultra low duty cycle (≪ 0.01%) without any problem such as high energy consumption. WRP has distinct features from existing routing protocols for WSN. At first, sensor nodes except a sink node require very small memory thanks to centralized routing protocol. Second, WRP has very low control message overhead because it does not rely on hello messages to check neighbor nodes or forwarding node’s aliveness. Last, WRP does not use any flooding to recover routing. WRP uses multi-hop wakeup time estimation based on one-hop wakeup time estimation information maintained internally by W-MAC and this enables routing recovery without high energy consumption.

Before detailed description of WRP, we assume:

- All nodes except a sink node have a same duty cycle and a same wakeup interval.
- Source nodes periodically send data packets to a sink node.
- WRP runs on W-MAC.

We show WRP’s operations and its multi-hop wakeup time estimation.

A. WRP operation

WRP is composed of 3 phases such as network setup, network start and network maintenance. Network setup and start phase are needed for network initialization and WRP runs only in network maintenance phase after initialization has finished.

1) Network setup phase: When WSN is deployed, all nodes are initially working in a high duty cycle mode and each node finds its neighbor nodes by broadcasting packets. After deployment, a sink node floods network setup messages. Whenever each node receives this message, it selects its parent node, that is a next-hop node for data packet forwarding like [2] and informs the parent node that this node becomes its child node. Then, each node sends a neighbor node list and a parent node ID (Identification) to a sink node. This phase finishes when the sink node receives messages from all nodes.

2) Network start phase: After the network setup phase is completed, the sink node broadcasts a network setup message. When each node receives a message, node randomly chooses its first wakeup time in the ultra low duty cycle mode and informs child nodes of the selected time. This time information is important because each child node should wake up for one wakeup interval at worst case to find a wakeup time of a parent node if it does not have this information. Then, each node runs on the ultra low duty cycle mode.

3) Network maintenance phase: In network maintenance phase, the sink node collects the wakeup time information of all nodes and it performs the routing recovery when a routing path is broken. For this, each node periodically sends its wakeup time information to a sink node every T sec. This information can be embedded into a normal data packet or carried by a control message called synchronization (SYNC) message. To avoid increasing number of control messages, the SYNC message is generated by leaf nodes which are not source nodes. Multiple wakeup informations can be aggregated and embedded into one packet unless increased packet size exceeds the maximum packet size. The wakeup time information consists of 5 fields as follows:

- \( i \): ID of the node that makes the information for its wakeup time estimation.
- \( D_i^j \): elapsed time after last wakeup of node \( i \) when node \( j \) receives a SYNC message.
- \( e_i^j \): the ratio of a node \( j \)'s clock speed to that of node \( i \).
- \( \theta_i^j \): the ratio of a margin rate for a error caused by a relative clock drift of node \( i \) and \( j \).
- \( d_i^j \): the ratio of a margin for a error caused by a non-clock drift of node \( i \) and \( j \).

where node \( j \) is included in the path between node \( i \) and a sink node or node \( j \) is the sink node itself.

When node \( j \) sends a packet which contains wakeup time information to its parent node whose ID is \( k \), node \( j \) should update \( D_i^j, e_i^j, \theta_i^j \) and \( d_i^j \) in the packet into \( D_i^k, e_i^k, \theta_i^k \) and \( d_i^k \). Detailed update algorithms are shown in Section III-B.

Every parent node receives data or SYNC message packets from all child nodes. Since the inter-packet time of these packets can not be larger than \( T \), parent node can acknowledge the child node’s failure if no packet is arrived during \( \alpha T \) where \( \alpha > 1 \).

When a parent node detects its child node’s breakdown, it sends a link error message to the sink node to trigger routing recovery. When the sink node receives this message, it searches a new routing path by using a network information. If found, the sink node sends a routing recovery message. This message contains a new path and wakeup time estimation informations for new links. If the new path contains new links, parent nodes of the new links do not have the information of the child nodes’ wakeup times, which belong to the new links. It means the parent node should find child node’s wakeup time during wakeup interval at worst case and it can cause a high energy consumption. To avoid this problem, the sink node provides its wakeup time estimation informations for new child nodes to parent nodes.

The routing recovery packet is forwarded according to the reverse order of the new path and each node which receives this packet should update its parent node and child node informations if necessary.

B. Multi-hop wakeup time estimation

Multi-hop wakeup time estimation algorithm is used to update wakeup time information fields in a message during transmission of a SYNC message and a routing recovery message. Basic concept of multi-hop wakeup time estimation is extension of one-hop wakeup time estimation. Thanks to this, the sink node can know all nodes’ wakeup time information and it can also inform this information to a node which wants to know a wakeup time of its neighbor node.

\(^2\)The farthest node from a sink node in a path.

\(^3\)One-hop closer node to a sink node in the path.
For multi-hop extension, we re-define \( n \) should transmit a new packet to node \( n \) to each other anymore. A and node 1, node 2, ..., node \( n \) are transferred from node 0 and it is transferred from node 0 to a sink node via message. This process finishes when the sink node receives the SYNC message whenever it is transmitted to a parent node.

When node \( n \)’s successful packet transmission time to node \( p \) is \( t_{\text{last}} \), wakeup interval is \( W \), the current time is \( t \) and node \( n \) should transmit a new packet to node \( p \), then the RX node’s wakeup time, \( \tau \), can be obtained as

\[
\tau = \left\lceil \frac{t - t_{\text{last}}}{W} \right\rceil \cdot \frac{W}{e_n^0} + t_{\text{last}},
\]

where \( \lceil \cdot \rceil \) is ceiling operator and its margin is given as

\[
M = (\tau - t_{\text{last}}) \cdot \theta_n^0 + d_n^0.
\]

As a result, node \( n \) needs to wake up only during \( \tau - M \sim \tau + M \) to find the RX node’s wakeup time.

2) Multi-hop wakeup time estimation for SYNC message:

For multi-hop extension, we re-define \( e_i^1, \theta_i^0 \) and \( d_i^0 \) as the multi-hop wakeup time estimation information fields. Therefore, node \( i \) and node \( j \) do not need to be a neighbor node to each other anymore. And \( D_i^0 \) means the elapsed time after the last wakeup time of node \( j \) when node \( i \) receives a SYNC message based on node \( i \)'s clock. Multi-hop wakeup time estimation information is calculated and saved into a SYNC message whenever it is transmitted to a parent node. This process finishes when the sink node receives the SYNC message.

From Fig. 1, we assume that a SYNC message is generated by node 0 and it is transferred from node 0 to a sink node via node 1, node 2, ..., node \( n - 1 \) in sequence. At first, node 1 obtains \( e_0^0, \theta_0^0, d_0^0 \) and \( D_0^0 \) when it receives a SYNC message from node 0. When node 2 receives the SYNC message from node 1, it can calculate new values using the old values in the message.

\[
\begin{align*}
    e_0^0 &= e_1^0 \cdot e_2^1, \\
    \theta_0^0 &= \theta_1^0 + \theta_2^1, \\
    d_0^0 &= (d_1^0 + d_2^0 + \theta_2^1 \cdot \Delta_2^1) \cdot e_2^1, \\
    D_0^0 &= (D_1^0 + \Delta_2^1) \cdot e_2^1,
\end{align*}
\]

where \( \Delta_2^1 \) is a time difference of a SYNC message reception time of node \( x \) and that of node \( y \), which is a neighbor node of \( x \). \( \Delta_2^1 \) is given based on node \( y \)'s clock. Fields of (3) are updated and saved into the SYNC message by node 2 again.

From (3), we can know that each field can be determined by the the SYNC message fields and the RX node’s information only.

If a path between node 0 and the sink node whose ID is \( n \) is represented as \( \{0, 1, 2, ..., n\} \), where node \( l + 1 \) is a parent of node \( l \), where \( 0 \leq l < n + 1 \) like Fig. 1, finally, the sink node gets the wakeup time information of node 0 as follows.

\[
\begin{align*}
    e_n^0 &= e_{n-1}^0 \cdot e_{n-1}^1 = \prod_{k=1}^{n} e_k^{k-1}, \\
    \theta_n^0 &= \theta_{n-1}^0 + \theta_n^{n-1} = \sum_{k=1}^{n} \theta_k^{k-1}, \\
    d_n^0 &= (d_{n-1}^0 + d_n^{n-1} + \theta_n^{n-1} \cdot \Delta_n^{n-1}) \cdot e_n^{n-1}, \\
    D_n^0 &= (D_{n-1}^0 + \Delta_n^{n-1}) \cdot e_n^{n-1}.
\end{align*}
\]

With the same way, the sink node has wakeup time informations of all nodes.

When the sink node receives a SYNC message from each node, it saves these results and its received time, which is needed to calculate \( \Delta_{n+1}^n \). From this information, the sink node can estimate the wakeup time of any node and calculate the margin. This information is updated whenever the sink node receives a new message.

3) Multi-hop wakeup time estimation for routing recovery:

As we mentioned in Section III-A3, a parent of a failed link or a node should send a link error message to a sink node and the sink node sends a routing recovery message to establish a new path. Routing recovery always needs a new link connection. But the parent node of the new link has no information about when child node becomes awake.

This problem can be solved by using the multi-hop wakeup time estimation information maintained by the sink node. When the sink node sends a routing recovery message, the message contains the wakeup time estimation information for the child node of the new link. The information should be updated whenever the message is forwarded and this process is the same with the way explained in Section III-B2. For example, from Fig. 1, let’s assume that node 1 failed and node 2 sent a link error message to the sink node. The sink node found a new path \( \{0, m, m-1, ..., n + 1, n\} \). For this path, a new link from node 0 to node \( m \) should be created. When node \( m \), the parent node of the new link, receives the routing recovery packet, it can obtain the wakeup time information of node \( m \).

\footnote{It is a parent node of node \( i \) in the routing layer.}

\footnote{Node \( j \) should be a parent or child node of node \( i \) in W-MAC.}

\footnote{Detailed description is omitted due to the lack of space.}

\footnote{The node closer to the sink node among two nodes of a link in a routing path.}
node 0, the child node of a new link, as follows.

\[ e_{m}^{0} = e_{m}^{0} \cdot e_{m}^{m-1} \cdot \prod_{k=n+1}^{m} e_{m}^{k-1}, \]

\[ \theta_{m}^{0} = \theta_{m}^{0} + \sum_{k=n+1}^{m} \theta_{m}^{k-1}, \]

\[ \phi_{m}^{0} = \phi_{m}^{0} + \phi_{m}^{m-1} \cdot \Delta_{m}^{m-1} \cdot e_{m}^{m-1}, \]

\[ D_{m}^{0} = (D_{m}^{0} + \Delta_{m}^{m-1}) \cdot e_{m}^{m-1}. \]

With this information, node \( m \) can calculate the wakeup time of node 0 similarly with (1) as follows.

\[ \tau' = \left[ \frac{D_{m}^{0}}{\theta_{m}^{0}} \cdot W \cdot \frac{W}{e_{m}^{0}} + t - D_{m}^{0} \right], \]  

where \( t \) is the current time.

And the margin is given as

\[ M' = (D_{m}^{0} + \tau - t) \cdot \theta_{m}^{0} + d_{m}^{0}. \]  

As a result, node \( m \) will wake up only during \( \tau - M' \sim \tau + M' \) to find a node 0’s wakeup time.

IV. PERFORMANCE EVALUATION

We used ns-2 simulator to evaluate the performance of WRP. Two duty cycle configurations are used for a low duty cycle and a ultra low duty cycle as in Table I. Two network topologies are used to simulate low and high node densities. The network sizes are 200 m \( \times \) 200 m for low node density and 100 m \( \times \) 100 m for high node density. For all cases, 25 nodes are used and they are randomly deployed. We compared energy consumptions of 3 routing protocols and routing recovery overheads of 2 protocols as follows.

- Static : only packet forwarding without routing maintenance overhead. It is ideal in terms of energy consumption.
- WRP : packet forwarding with routing maintenance overhead by our proposed protocol, i.e., WRP.
- FLOOD : packet forwarding with periodic control message exchange with neighbor nodes.
- WRP+R : packet forwarding and routing recovery by WRP. During simulation, one randomly selected node is failed.
- FLOOD+R : packet forwarding with periodic control message exchange and routing recovery by flooding. During simulation, one randomly selected node is failed.

FLOOD is not a specific name of a routing protocol and it denotes a generic routing protocol which can be on-demand routing protocols like AODV [5] or any proactive routing protocols if it uses packet flooding for routing recovery. For any cases, each node should exchange messages periodically with neighbor nodes to maintain the wakeup time information of all neighbor nodes. This information is needed because packet flooding assumes that each node can communicate with any neighbor nodes and it means each node should know the wakeup time information of all neighbor nodes. We call this message as SYNC message without confusion. We ignore additional energy consumption caused by other operation of specific routing protocol.

For SYNC message of FLOOD, the transmission interval is set to 2 times of the data traffic interval. For SYNC message of WRP, the transmission interval, \( T_s \), is also set to 2 times of the data traffic interval and \( \alpha \) is set to 2.

A. Normal energy consumption

From Figs. 2 and 3, the increment of consumed energy of WRP is very small and it is almost same with a static routing, which is the best for energy consumption due to no control overhead. The maximum performance improvement is about 32% for low node density, and 67% for high node density when no routing recovery is occurred. These results come from the comparative routing protocol’s overhead caused by the synchronization between each neighbor node.

WRP does not need any synchronization with neighbor nodes at all and most of control informations for routing maintenance are carried by normal data packets. These features enable WRP to minimize the control overhead. From Figs. 2 and 3, FLOOD consumes energy about 200% \( \sim \) 1000% more than a static routing. The energy consumption of this protocol increases proportionally to the number of neighbor nodes. Due to the reason, WRP outperforms others more and more if node density increases.

Generally, the energy consumption should be increased if the duty cycle rate becomes low. This increased energy consumption is caused by the synchronization error due to a clock drift. As a result, previous protocol’s energy consumption becomes increased from 70% to 320% for low node density and 230% to 1030% for high node density in comparison with Static. On the contrary, for WRP, the energy consumption is almost same even though the duty cycle is changed. Figs. 2 and 3 show only 21% increment for low node density and 8% increment for high node density in comparison with Static.

B. Energy consumption of routing recovery

To measure the energy consumption caused by routing recovery, one node is selected randomly and it stops its operation when simulation time reaches 20% of the total simulation time. FLOOD protocol should flood control packets in order to find a optimal path and it consumes huge amount of energy in a short time. From Figs. 2 and 3, we can see that energy consumption caused by routing recovery can be increased when either duty cycle or node density is low. We already explained that the energy consumption caused by control packets is increased because of the increased clock drift in Section IV-A. For high node density, each node has many neighbor nodes and it causes many SYNC message exchanges and due to this high cost, the routing recovery cost becomes relatively small. On the
contrary, the routing recovery cost dominates over other costs for low node density. This explains why the routing recovery cost of FLOOD becomes critical when node density is low.

For WRP, the routing recovery cost is very small thanks to unicast transmission of a routing recovery packet and multi-hop wakeup time estimation algorithm. These minimize the routing recovery costs for all cases. The performance gap becomes larger when a duty cycle is low and node density is high. WRP consumes 53% energy consumption of FLOOD for a low duty cycle and low node density and only 7.9% for a ultra low duty cycle and high node density.

V. CONCLUSION

Our proposed WRP is a centralized routing protocol that is designed to run for low to ultra low ($\ll 0.01\%$) duty cycled networks. The sink node in WRP uses the estimated information about each node’s wakeup time to minimize energy consumption. We enhanced the previous wakeup time estimation algorithms designed for one-hop communication in W-MAC [1] to cover the multi-hop communication. This feature enables routing recovery to be feasible with minimal energy consumption when some links are broken. In WRP, the memory requirement at the sensor nodes and the overhead for control messages are very small. Overall, WRP is a promising routing protocol for applications like monitoring which needs long network lifetime under a loose delay constraint.

ACKNOWLEDGMENT

This research is supported by the Ubiquitous Computing and Network (UCN) Project, Knowledge and Economy Frontier R&D Program of the Ministry of Knowledge Economy (MKE) in Korea as a result of UCNs subproject 09C1-C1-20S.

REFERENCES