Wireless Netw (2014) 20:833–846
DOI 10.1007/s11276-013-0645-2

Opportunistic routing with in-network aggregation for asynchronous duty-cycled wireless sensor networks

Jungmin So · Heejung Byun

Published online: 4 October 2013
© Springer Science+Business Media New York 2013

Abstract We propose an opportunistic routing protocol for wireless sensor networks designed to work on top of an asynchronous duty-cycled MAC. Opportunistic routing can be very effective when used with asynchronous duty-cycled MAC because expected waiting time of senders—when they stay on active mode and transmit packet streams—is significantly reduced. If there are multiple sources, energy consumption can be reduced further through in-network aggregation. The idea proposed in this paper is to temporarily increase duty cycle ratio of nodes holding packets, in order to increase chance of in-network aggregation and thus reduce energy consumption and extend network lifetime. In the proposed protocol called opportunistic routing with in-network aggregation (ORIA), whenever a node generates a packet or receives a packet to forward, it waits for a certain amount of time before transmitting the packet. Meanwhile, the node increases its duty cycle ratio, hoping that it receives packets from other nodes and aggregate them into a single packet. Simulation results show that ORIA saves considerable amount of energy compared to general opportunistic routing protocols, as well as tree-based protocols.

Keywords Wireless sensor networks · Opportunistic routing · In-network aggregation

J. So
Department of Computer Engineering, Hallym University, Chuncheon, Korea
e-mail: jso1@hallym.ac.kr

H. Byun (✉)
Department of Information and Telecommunications Engineering, Suwon University, Hwaseong, Korea
e-mail: heejungbyun@suwon.ac.kr

1 Introduction

Energy efficiency is a well-known key factor in designing wireless sensor networks. The major goal of MAC and routing protocols is to put nodes into sleep mode as much as possible, because energy consumption in sleep mode is significantly lower than active mode [1]. Duty-cycling, in which nodes periodically switch between active and sleep mode, is a most widely used method for saving energy. Duty cycle is the percent of time a node spends in an active mode as a fraction of total time. A node can transmit or receive packets while in active mode, but cannot communicate while in sleep mode. Thus, increasing duty cycle results in higher energy consumption but lower delay, while decreasing duty cycle produces an opposite effect.

MAC protocols using duty-cycling can be classified into two categories: synchronized and asynchronous. A synchronized MAC protocol synchronizes the active periods of nodes so that they wake up at the same time. In some protocols all nodes wake up at the same time, whereas in other protocols node pairs in need of communication are scheduled to wake up at the same time. Either way, a sender can believe that the receiver is in active mode when it transmits a packet. The problem with synchronized protocols is that nodes need to be kept time-synchronized. Control packets need to be transmitted periodically in order to make the nodes synchronized, which is an overhead required for these protocols. Also, if nodes become out of sync, more energy is needed to restore the synchronization again (such as occasionally having all nodes wake up for a certain period of time). Asynchronous protocols do not synchronize active periods, but makes the sender wait for the receiver in active mode, until the receiver wakes up. For example, if node A has a packet to transmit to node B, node A wakes up and continuously transmits its packet, until node B wakes up and receives the packet. When node B receives the packet, it sends back an ACK so that node A can stop transmitting the packet and go back to sleep. Asynchronous protocols are simple to implement and operate, one of these protocols, BoX-MAC [19] has been selected as a default MAC protocol in TinyOS 2.×. More details of existing synchronized and asynchronous protocols will be discussed later in the paper.

A routing protocol, that runs on top of a MAC protocol, is designed also with consideration for energy consumption. Routing also depends on data collection pattern of an application. Here, we consider a typical sensor network application where sensor nodes send their readings to the sink, if a certain conditions are met. A typical routing strategy is to build a tree rooted at the sink node, and packets are forwarded along the branches of the collection tree. However, if we consider asynchronous duty-cycling at the MAC layer, an opportunistic routing approach can significantly reduce energy consumption. In an opportunistic routing scheme, a node has multiple potential forwarders instead of just one (the parent) as in a tree-based routing. The node sends its packet to whoever wakes up first, in order to reduce delay and also energy consumed waiting for the receiver to wake up. ORW [3] is the protocol that uses this approach and is shown to save a significant amount of energy compared to a tree-based routing.

The problem with ORW is that it does not consider the benefit of in-network aggregation. Suppose there are multiple sources that generate packets at a similar time. When a node tries to send its packet to one of the forwarders, some of them may already have other packets that are waiting to be sent. In this case, it is better to send the packet to that particular forwarder which already holds a packet, since the packets can be aggregated into one packet. In-network aggregation is a technique used in wireless sensor networks to reduce number of packet transmissions [4]. The size of a packet after aggregation depends on the application. If the application is looking for the minimum or the maximum, the packet size does not have to increase. If the application is looking for an average, the packet needs to contain the total value and the number of data aggregated. Thus the packet size does not grow with the number of aggregated packets for average as well. If the application needs to see the data separately, in-network aggregation is still beneficial because the preamble and the packet header occupy a significant portion of a packet. Even if multiple data are concatenated, the packet size does not grow linearly. In addition, overhead of transmitting a packet, such as waiting time of the sender in a duty-cycled MAC can be saved by in-network aggregation. In this paper, we assume perfect aggregation, which means we neglect the packet size growth after aggregation.

If a node can forward its packet to a forwarder which already holds a packet, energy consumption can be reduced using in-network aggregation. However, the node does not know which one of the forwarders is holding a packet. If the packet sources and their routes to the sink are fixed as in a tree-based protocol, this may be possible. But if packet sources change dynamically and the routes are not fixed as in an opportunistic routing, this knowledge is not given. The motivation of this paper is simple: we would like to increase chance of packets being aggregated at intermediate nodes, so that the energy is saved by reducing packet transmissions. The proposed idea is to control duty cycle of nodes depending on whether it has a packet to send or not, so that nodes holding packets have a higher chance of receiving packets from downstream nodes. This simple idea can lead to considerable amount of energy savings, as shown in the performance evaluation section.

The rest of the paper is organized as follows. In Sect. 2, we summarize and discuss some relevant related work. In Sect. 3, we provide preliminary background on asynchronous duty-cycled MAC protocols and opportunistic routing protocols. Also, we describe the network and application model we use and motivate the problem. In Sect. 4, we present our proposed protocol, opportunistic routing with in-network aggregation (ORIA), which controls duty cycle of nodes based on whether they have packet to send or not, in order to promote in-network aggregation and extend network lifetime. In Sect. 5, we evaluate performance of the proposed protocol, in comparison with tree-based routing protocols and opportunistic routing protocols. Finally, we conclude with remarks on future work in Sect. 6.

2 Related work

Huge amount of research efforts have been put into wireless sensor networks during the last decade. Although similar to wireless multi-hop networks such as ad-hoc networks, wireless sensor networks have unique characteristics that make protocol design different from other types of wireless networks [5]. The most important characteristic is that nodes are energy limited, and typically expected to run without human intervention for a very long time. Thus, energy efficiency is the most significant factor in making design choices. Traffic pattern is another thing that is unique to the sensor networks. In typical applications traffic load is low, and the traffic is directed towards the sink or multiple sinks. In most cases, prolonging network lifetime is preferred to increasing reliability or reducing packet delay. Although not considered typical, there are applications that require high traffic load, such as wireless multimedia sensor networks [6]. Also, quality of service requirements may exist, such as reliability or delay [7].

Medium access protocols for wireless sensor networks can be divided into synchronized and asynchronous protocols. A synchronized protocol synchronizes active times of sender and receiver, so that the receiver is awake when the sender transmits its packet. While there are many ways of synchronizing sender and receiver, using common active periods is a well-known approach. SMAC [9] is the basic protocol in this category. In SMAC, all nodes in the network wake up at the same time and stay in active mode for a fixed duration of time. Time is synchronized among nodes by exchanging SYNC messages at the beginning of an active time. To avoid collision, the RTS/CTS mechanism is used. TMAC [10] is an improved version of SMAC, in which the duration of active period adaptively shortened if there is no more traffic. In DSMAC [11], the duration of active period is adaptively increased and decreased based on delay requirements and traffic load. Many different protocols build upon these basic schemes to minimize energy consumption using common active periods [12–16].

Asynchronous protocols do not synchronize active times of sender and receiver, but let the sender wait for receiver to wake up while transmitting on the channel. B-MAC [17] is the basic protocol, where the sender sends a preamble on the channel for the duration of a whole wakeup period, and nodes that receive the preamble stays up until the sender sends a packet. In X-MAC [18], the sender sends short preambles, with a time gap between two preambles where a receiver can send back an ACK. The preamble includes the destination address, so nodes other than the destination can go back to sleep right after checking the preamble. In BoX-MAC [19], the sender sends the entire packet repeatedly, instead of preambles. When the receiver sends back an ACK, the communication is finished without needing another exchange of packets. Other protocols exist that follow the line of asynchronous wakeup [20–22].

Routing protocols run top of MAC protocols and decide the paths where packets are forwarded. Protocols designed for multi-hop ad hoc networks or mesh networks may work in wireless sensor networks, but typically they are not suitable because they are not designed for energy efficiency, and they do not consider the benefit of in-network aggregation. Krishnamachari et al. [4] proposed three mechanisms for selecting routes considering aggregation. Among the schemes, the greedy incremental tree (GIT) provides the minimum number of transmissions, thus minimum energy consumption. Aonishi et al. [23] proposed a new metric for selecting routes considering aggregation efficiency. The idea is that if aggregation efficiency is low, it would be better to use the shortest path instead of the path that leads to better aggregation. Mottola et al. proposed the Muster protocol [24], which is a routing protocol designed for multi-sink environment where packets from a source are forwarded to multiple sinks. The aim is to maximize aggregation, so that number of transmissions is minimized. Liu et al. [27] proposed a routing protocol with in-network aggregation, where they have considered what they call the aggregation ratio. The aggregation ratio is the maximum number of packets that can be aggregated into a single packet. If the aggregation ratio is infinite, it is the perfect synchronization. The routing strategy when aggregation ratio is finite is to choose a parent node with a packet that has room left for more packets to be aggregated. [27] assumes that all nodes in the network are packet sources, and thus the best strategy is to build an aggregation tree that balances the number of descendants among siblings. Villas et al. proposed DRINA, a tree-based routing protocol designed to achieve maximum benefit from in-network aggregation [28]. In DRINA, multiple nodes sensing an event form a cluster by electing a cluster head and send their packets to the cluster head for aggregation. Also, the cluster heads are connected to the sink by a tree. Initially, a tree rooted at the sink is built in the network, and the hop distance of a node is the distance between itself and the sink. When an event occurs, the path from the cluster head to the sink becomes an "established route". The hop distance of all other nodes are updated so that their hop distance is the shortest distance between itself and a node in the established route. Using this hop distance metric, a GIT-style structure is built in the network, which is shown to perform well with in-network aggregation. These routing protocols are designed to minimize packet transmissions using by promoting in-network aggregation, but they do not consider opportunistic routing. With opportunistic routing, packets are not forwarded along an established tree, and the packet aggregation will not happen if packet follow different paths. These protocols have no mechanism to improve chance of aggregation when opportunistic routing is used.

Opportunistic routing protocols have been proposed in the context of ad-hoc networks and sensor networks. In ExOR [2], the sender includes a prioritized list of forwarders in the packet header. When a node receives the packet, it forwards the packet after a certain delay computed from the priority of itself in the list of forwarders. Other nodes who overhear the packet transmission cancels its transmission, thus avoiding redundant forwarding. This protocol is not suitable for wireless sensor network, since nodes cannot overhear packet transmissions while sleeping. GeRaF [25] decides the forwarder list based on the location information. Nodes that are geographically close to the destination becomes the forwarder. To avoid multiple nodes forwarding the same packet, busy tone is used. GeRaF has similar problems with ExOR when applied in duty-cycled wireless sensor networks. Gu et al. proposed DSF [26], an opportunistic routing protocol designed to work with synchronized MAC. Forwarding nodes are selected based on sleep schedules as well as delay, reliability and energy consumption. The protocol requires control message overhead in order to distribute sleep schedules of nodes. Aitsaadi et al. proposed three variants of opportunistic routing protocols for duty-cycled MACs [29]. The first one, called Basic-opportunistic, forwards the packet to the node that wakes up first, if the node is geographically closer to the sink than the sender. The second one, called Opportunistic with delay, limits the waiting time of the sender. The packet is discarded if a time limit is reached, where in Basic-opportunistic the packet may wait for an unlimited time. The time limit is used to avoid draining the energy of a node waiting for the receiver, sacrificing reliability. The third one, called Opportunistic with backtracking, allows a packet to move further away from the sink, if there is no path that moves toward the sink. This scheme is proposed to avoid route holes in the network. Landsiedel et al. have proposed ORW [3], which is also targeted for asynchronous duty-cycled networks. Similar to [29], ORW selects the first node that wakes up as the forwarder, among a set of candidate forwarders. The candidate forwarders are selected based on a metric called EDC (Expected Duty Cycled Wakeups), which is based on number of neighbors and link quality. Since ORW is used as a ground for our proposed protocol, we discuss ORW in more detail in the next section. These opportunistic routing protocols can reduce significant amount of energy compared to unicast routing protocols. However, they both do not consider benefit of in-network aggregation when selecting the forwarder. Considering in-network aggregation can further reduce energy consumption, especially when packets are generated from multiple nodes. It is the aim of our proposed protocol.

3 Preliminaries

3.1 Asynchronous duty-cycled MAC protocols

B-MAC [17], X-MAC [18], and BoX-MAC [19] are the most widely known asynchronous MAC protocols. Their common feature is that if a node has a packet to send, it wakes up and waits for the receiver to wake up, while continuously transmitting something on the channel. Otherwise, nodes wake up according to their duty cycles, unsynchronized with their neighbors.

In B-MAC, the sender transmits a long preamble while waiting for the receiver. When a node wakes up according to its schedule, it listens to the channel to see if there is any preamble being transmitted. If there is a preamble, it stays up. If not, the node goes back to sleep. The sender transmits the preamble long enough so that all of its neighbors are awake when the sender transmits the data packet. If the duration of a wakeup period (a duration of time between two active modes) is the same for all nodes, the sender must transmit the preamble for the duration of a whole wakeup period. If wakeup periods are different, the sender must transmit the preamble for the duration of the longest wakeup period. Followed by the preamble, the sender transmits the data packet which should be received by the receiver, since the receiver will be awake. B-MAC has two problems that lead to waste of energy. First, even if the receiver wakes up shortly after the sender starts transmitting preamble, both the sender and the receiver should wait in active mode until the end of the preamble. Second, nodes that are not the destination of the packet should also be awake until the packet is transmitted.

X-MAC addresses the two problems of B-MAC using short preambles. Instead of sending a long preamble, the sender transmits a stream of short preambles that include the destination address. When a node wakes up according to its schedule, it listens on the channel for a duration time enough to receive a short preamble. If a preamble is received, the node checks to see if the packet is destined for the node. If the node is not the destination, it goes back to sleep. If it is indeed the destination, it sends an acknowledgement (ACK) packet to the sender. Then the sender sends the data packet, and the receiver replies back with an acknowledgement for the data packet. BoX-MAC is slightly different from X-MAC. In BoX-MAC, instead of sending preambles, the sender sends the packet itself in streams. This removes one of the two handshakes in X-MAC. Also, nodes waking up do not wait for the whole packet duration, but checks the channel to see if there is energy on the channel, and go back to sleep if no energy is detected. This reduces the wakeup duration for nodes when there is no transmission taking place. Specifically, a node running X-MAC should stay up for 20ms to find out that it can go back to sleep, while a node running BoX-MAC only needs to stay up for 5.61ms [19].

Figure 1 describes the behavior of BoX-MAC.

3.2 Opportunistic routing protocols

The general concept of opportunistic routing is that a node can have multiple potential receivers instead of a single destined receiver. The node forwards its packet to one of the receivers who is in the best condition. [2]. Protocols using opportunistic routing in wireless networks often selects the receiver that has the best link condition at the point of packet transmission, increasing packet delivery ratio and reducing delay.

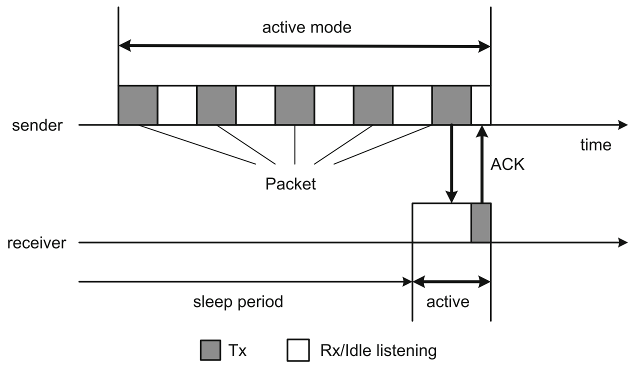

Fig. 1 Behavior of BoX-MAC. The sender transmits *packets* in streams, until the receiver wakes up and receives the packet. On receiving the *packets*, the receiver sends back an ACK and goes back to sleep. The sender also goes back to sleep when it receives the ACK

When an opportunistic routing is combined with asynchronous duty-cycled MAC, the routing protocol can choose the receiver that wakes up first. This reduces packet delay, but what is more important is that it can save significant amount of energy that the sender consumes waiting for the receiver to wake up. Protocols proposed in [29] and [3] use this approach, and is shown to outperform tree-based routing protocols in terms of energy consumption.

The behavior of an opportunistic routing protocol is illustrated in Fig. 2. Suppose node B and C are candidate forwarders of node A. When node A has a packet to send, it wakes up and starts sending its packet in streams. Suppose by chance node B wakes up earlier than C. When node B receives the packet it sends back an ACK to node A which can finish transmission and go back to sleep. If C wakes up first, C will also accept the packet from A. Thus, if there are N forwarders, the expected waiting time for the sender is reduced to $1/N$.

The candidate forwarders of a node can be chosen in various ways. The simplest method is to include all neighbors that have shorter hop distance to the sink in the forwarder set. To implement this, each node only needs to know its hop distance to the sink. A node does not even need to know who are the neighbors, unless its hop distance is not changed. The problem with this method is that if the packet is forwarded along nodes with very few number of neighbors, the total energy consumed for delivering this packet to the sink may be large. Also, it may be beneficial to send the packet to a node with the same or even larger hop distance than the sender, depending on the expected wait time of these nodes. Thus, ORW [3] proposes a metric call EDC (Expected Duty Cycled Wakeup), which considers the number of neighbors when selecting the forwarder set. (ORW also considers link delivery ratio, but it is omitted here for simplicity.)

EDC of a node is calculated as follows. For a subset S_i of node i, EDC_i is:

$$EDC_i(S_i) = \frac{1}{|S_i|} + \frac{\sum_{j \in S_i} EDC_j}{|Si|} \tag{1}$$

The first term is the expected wait time of the node itself, and the second term is the average expected wait time after the node is forwarded. Initially, S_i is empty. For each iteration, a neighbor j with the lowest EDC is inserted into S_i, and $EDC_i(S_i)$ is updated. This iteration stops when EDC_j becomes larger than $EDCi(S_i)$. As shown in Sect. 5, using EDC improves network lifetime compared to just using hop distance. However, in an environment where network topology changes frequently, nodes need to exchange messages with neighbors to constantly update the EDC. In this paper we consider both methods of selecting candidate forwarders.

Without in-network aggregation, there is no need for priority among nodes in the forwarder set. A node can just forward its packet to the node that wakes up first. However, if some of the nodes in the forwarder set already have a packet waiting to be sent, there is a chance for further energy saving by forwarding the packet to one of these nodes. This observation motivates an opportunistic protocol that is designed to benefit from in-network aggregation.

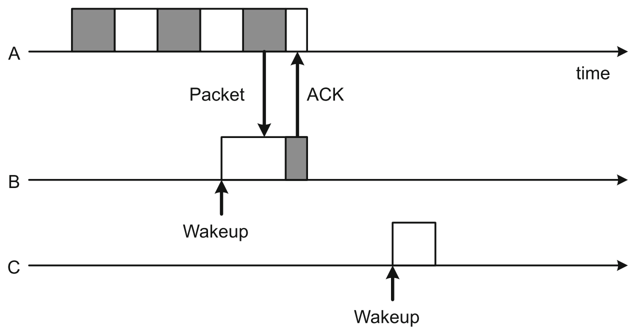

Fig. 2 Behavior of an opportunistic routing protocol with duty-cycled MAC. In this example nodes *B* and *C* are As candidate forwarders. Whoever wakes up first receives the packet and sends back an ACK back to the sender

3.3 Network model and motivation

We consider a wireless sensor network with N nodes. One of the node is the sink node, which serves as a data collection point. We focus our attention to a single sink case, although the proposed idea can be directly applied to a multi-sink environment. Once deployed, each node periodically generates a packet if a certain criteria is met. Thus, only a portion of nodes may become packet sources, and they can change over time. An example application that has this kind of traffic pattern is temperature and humidity monitoring in a building, where nodes are configured to send periodic reports when a certain criteria is satisfied. On generating a packet, the node forwards the packet to the sink using opportunistic routing.

Now consider the scenario in Fig. 3. At a period, nodes A and B become packet sources because their sensor readings meet the predefined criteria. Suppose node B, C, and D are the members of A's forwarder set. Using opportunistic routing, node A will send its packet to whoever wakes up first among B, C, and D. Suppose node C or D receives the packet. Then, the node should forward the packet to the sink. Meanwhile, node B also has to transmit, because it has a packet generated by itself. Now, suppose node B receives node As packet. Then, neither node C nor D needs to transmit a packet. B does not need to transmit twice, since the data can be aggregated into a single packet. (As mentioned earlier, we neglect the increase of packet size after aggregation.) Thus, in terms of total energy consumption, it would be a better choice if node A can forward its packet to node B, and not to C or D.

If packet sources are always the same for every period, we can build a collection tree considering in-network aggregation as in [4]. However, if packet sources change over time, it is not possible to predict who is the best next hop node in terms of total energy consumption. One possibility is to include a flag in the packet header. If the flag is set, only node that already have another packet receives the packet. If no node receives the packet for a certain duration of time, the sender can reset the flag so that any node can receive the packet. This will increase chance of packet aggregation, but average waiting time may increase depending on the number of source nodes. If there are small number of sources compared to the total number of nodes, this strategy will significantly increase waiting time and thus energy consumption.

In this paper, we propose to control the duty cycle of nodes in order to promote aggregation. When a node is holding a packet, it temporarily increases its duty cycle in order to increase the chance of receiving other packets. The details of the proposed protocol is described in the next section.

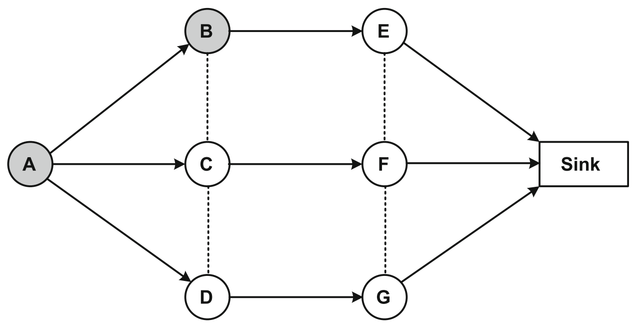

Fig. 3 A motivating example of the proposed scheme. If nodes *A* and *B* are packet sources, it is better for node *A* to send its packet to *B*, because packet aggregation leads to energy saving

4 The proposed protocol: ORIA

The proposed protocol is called ORIA. The basic operation of opportunistic routing is similar to ORW [3]. When a node sends a packet, it does not specify a destination. Instead, the packet header includes information on who can receive this packet. Any receiver who wakes up and receives the packet checks whether it should accept the packet or not. If the receiver accepts the packet, it sends an ACK back to the sender. The sender stops transmitting the packet once it receives the ACK. For selecting the forwarder set, we consider both schemes: hop distance-based and EDC-based.

4.1 Initial operation and updates

When nodes are deployed, they start their duty cycle by switching between active and sleep modes. The initial sleep period is set to t_s for all nodes. (Although the sleep period is the same, nodes wake up at different times since they are not time synchronized.) Then, the nodes need to find the hop distance to the sink. To find out its distance to the sink, a node starts transmitting a PATH-REQUEST message on the channel, as a stream of packets. A neighboring node wakes up according to its wakeup schedule and receives the packet. If the receiver knows its hop distance to the sink, it replies with an ACK which includes its hop distance. Once the sender receives an ACK, it stops sending PATH-REQUEST and goes back to sleep. If multiple nodes simultaneously receive the PATH-REQUEST message and send ACKs to the sender, The ACKs will collide. In this case, the sender will continue sending its PATH-REQUEST packet. If the sender does not receive an ACK for a certain amount of time, it stops transmitting, goes back to sleep and tries it again after some time. The node can also send a PATH-REQUEST during operation, if it fails to send its packet to any of the receivers. To use EDC, nodes periodically exchange HELLO message, including their EDCs in the packet. HELLO messages are broadcast, and they do not require ACKs.

4.2 Packet generation and forwarding

Once a node finds out its distance to the sink, it is ready to check sensor readings and generate packets if the conditions are met. (To use EDC, a node should find out the EDCs of its neighbors.) When a node generates a packet, it should forward the packet to the sink. In ORIA, the node does not immediately transmit the packet, but waits for a duration of time called *packet holding time*. During the packet holding time, the node performs *fast wakeup*, in which the node increases its duty cycle in order to increase chance of receiving packets from other nodes. An intermediate node who receives a packet from another node also goes through packet holding time before forwarding the packet. A node already in its packet holding time does not extend the packet holding time even if it receives a packet from a neighbor. The wakeup behavior of a node running ORIA is described in Fig. 4.

When the packet holding time is finished, the node checks the channel to see if it is idle, and transmits the packet in streams similar to the BoX-MAC. When using the hop distance metric, the sender includes its hop distance to the sink in the packet header. When using EDC, the forwarder set is included in the header. Neighboring nodes receive the packet once they wake up. If a node receives the packet, it looks at the header and see if it should receive the packet. When using the hop distance metric, the node accepts the packet if it has a shorter hop distance to the sink than the sender. Optionally, if the node has the same hop distance with the sender, it can still accept the packet if it is already holding another packet. If ORIA is using EDC as the metric, then the receiver simply accepts the packet if it is included in the forwarder set.

An example operation of ORIA using hop distance metric is shown in Fig. 5. In the figure, node T is the current packet sender. Nodes B and C (shown in gray filled circles) have packets to send, and are in the packet holding time. Nodes that receive T's packet are A, B, and C. Nodes A and B can accept the packet because they are closer to the sink, whereas node C can accept the packet because it has the same hop distance to the sink as the sender, and it is in packet holding time. Node D does not accept T's packet, even if it is the first node to wake up and receive the packet. If a node accepts the packet, it sends an ACK packet back to node T. On receiving ACK, node T stops transmitting packet streams and goes back to sleep.

Length of the packet holding time and the fast wakeup rate are parameters that can affect system performance. If the packet holding time is too short, the chance of aggregation will decrease. If it is too long, packet delay will be too high. Even if we consider applications with no real-time delay requirements, long packet delay often decreases the value of information. For applications with delay limits, the packet holding time can be set according to the requirements. If the fast wakeup rate is too low, it will reduce chance of aggregation. If the fast wakeup rate is too high, energy consumption will increase because nodes wake up too often. In Sect. 5, we evaluate the impact of these parameters on system performance.

4.3 Dealing with multiple node reception

The problem with opportunistic routing is that multiple nodes may receive the packet, if they are awake at the same time. If this happens, multiple nodes send ACKs to the sender, which will cause collision. The sender will continue transmitting its packet since no ACK is received. The problem becomes more severe because multiple nodes that received the packet will forward the packet anyway, causing redundancy and unnecessary energy consumption. Landsiedel et al. [3] suggests using probabilistic ACK transmission. When a node receives a duplicate packet, it sends an ACK with 50 % probability and forwards the

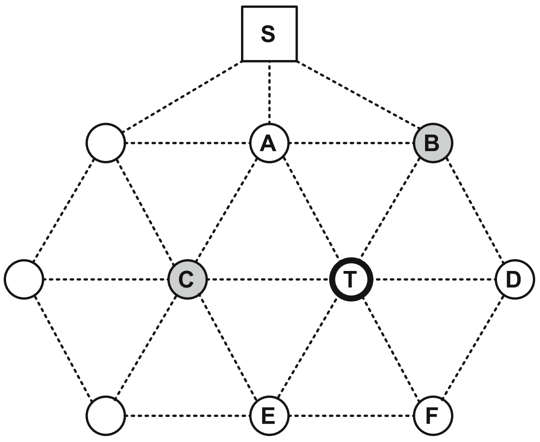

Fig. 5 An example scenario. *T* is the node transmitting packet. Nodes *B* and *C* are holding packets and are in the packet holding time. Nodes that can receive *T*s packet are nodes *A*, *B*, and *C*

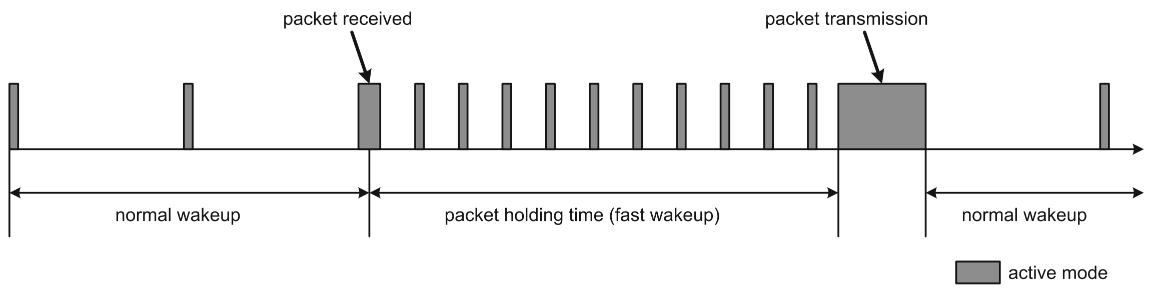

Fig. 4 The wakeup behavior of a node running ORIA. When a node receives a packet, it does not immediately send the packet, but waits for a duration called packet holding time. During this period, the node increases its duty cycle to receive packets from other nodes

packet only when it decides to send an ACK. Although this mechanism may reduce amount of redundant packets, it still produces a lot of duplicate packets. First, the multiple receivers still forward the first copy of packets that they receive. Also, as the number of neighbors increase, 50 % probability can still produce large number of ACK collisions. One can reduce the forwarding probability after each duplicate packet, but a lot of duplicates will be generated until only one packet is received at the sender.

In ORIA, we propose to use CONFIRM packets, which the data packet transmitter sends back to the receiver after receiving an ACK correctly. The CONFIRM packet confirms that the sender received a single ACK, and also specifies who should forward the packet to its upstream node. The CONFIRM packet includes the ID (or address) of the node who sent the ACK, telling the receiver to forward the packet to its upstream node (forwader set). This procedure is shown in Fig. 6. If an ACK collision occurs, the sender will not receive an ACK and thus does not send a CONFIRM packet. A node that receives a packet does not forward it unless a CONFIRM packet is received. This prevents redundant packet forwarding which causes significant waste of energy. When a node receives a data packet and does not receive a subsequent CONFIRM packet, it may still choose to keep the data for a certain duration. Later if the node has other packets to send, this data can be aggregated into the packet to increase reliability.

5 Performance evaluation

5.1 Simulation setup

Performance of the proposed protocol is evaluated using a simulator written in C++. We compare performance of the following six protocols:

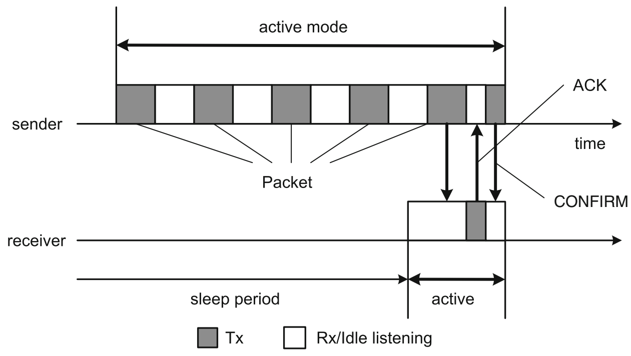

Fig. 6 The protocol behavior of *ORIA*. The sender sends packet streams until one of the potential forwarders receive the packet and replies with an *ACK*. Then, the sender sends a *CONFIRM* packet to the receiver indicating that the receiver is the only forwarder

1. TREE: This is a standard tree-based protocol, such as [8]. The packets are passed from the source nodes to the sink via established tree routes. For fair comparison, packets are aggregated if multiple packets happen to meet at a node.
2. TREE-HOLD: This is similar to TREE, but each node waits for packet hold time as in ORIA before transmitting the packet to its upstream node. Meanwhile, packets coming from downstream nodes are aggregated into a single packet.
3. OPPO: This is an opportunistic protocol that uses hop distance metric. When a sender transmits a packet, a neighboring node who is closer to the sink than the sender receives the packet and replies back with an ACK. Although proposed as a part of ORIA, the CONFIRM packet is also used in the OPPO protocol to reduce the negative effect of multiple node reception.
4. ORW: This is an opportunistic protocol proposed in [3]. Each node selects its forwarder set according to the EDC metric, and only nodes in the forwarder set receives the packet. CONFIRM packet is used here too, since how to deal with multiple node reception is not well defined in [3].
5. ORIA-HOP: This is the proposed protocol which uses the hop distance metric. Nodes hold on to the packets for packet hold time before transmitting, and meanwhile temporarily increase their duty cycle in order to increase chance of in-network aggregation. In this protocol, a node may receive the packet if it is closer to the sink than the sender, or it has the same hop distance with the sender but is holding a packet.
6. ORIA-EDC: This is the proposed protocol which uses the EDC metric. Only the nodes in the forwarder set receive the packet.

For simplicity, the link quality of all links is set to 1, which means no packet loss is occurred due to out-of-band interference or data packet collision. However, ACK collision is modeled in the simulation, which is considered a significant factor affecting the performance of opportunistic protocols. When the simulation starts, a number of nodes, as well as the sink node are randomly placed in a 100m x 100m region. The transmission range of a sensor node is 20 m, and the carrier sense range is 40 m, which is twice the transmission range. All nodes start with 2000mAh of energy which is a typical capacity of an AA-type rechargeable battery. The current consumption for Tx mode and Rx mode is 17.4 and 19.7 mA, respectively [1]. Also, the current consumption for switching between active and sleep mode is 0.3 mA [30]. When a node wakes up, it stays on the channel for 5.61ms and goes back to sleep if there is no signal on the channel. If there is a packet on the channel but the node decides not to receive it, the node stays in the active mode for 20ms. Finally, if the node receives the packet, it stays in the active mode for 50ms [19]. Once the nodes are deployed, control messages are exchanged in order to establish a tree rooted at the sink. The cost of establishing a tree, as well as the cost of updating routes or EDC, are not included in the energy consumption. Once the tree is established, nodes start generating packets. Unless otherwise specified, the packets are generated at random nodes according to the given packet generation rate. If the packet generation rate is 30 packets per 30 s, a packet is generated every 1 s at a random node. If a node generates a packet, it does not generate another packet for 30 s. The simulation ends when the first node runs out of energy, and the whole duration is considered as the network lifetime. Each point in the graphs is an average of results obtained from 50 different topologies and 20 different seed for each topology (a total of 1,000 runs).

5.2 Results

5.2.1 Impact of node density

In the first simulation, we compared the network lifetime of the protocols varying number of nodes. Since the size of the simulation area is fixed, number of nodes translates into node density. Normal wakeup time is 2 s, meaning each node wakes up every 2 s to check the channel. The boot up time of the nodes are randomized, so that the exact wakeup times vary among nodes. For TREE-HOLD, ORIA-HOP and ORIA-EDC, the packet hold time is 10 s, and the fast wakeup time is 500 ms.

The result is shown in Fig. 7. Figure 7(a) is the result when the packet generation rate is low, and Fig. 8(b) is the result when the packet generation is higher. As the node density increases, the network lifetime of protocols other than the tree-based protocols increase. The tree-based protocols do not benefit from increased node density since the routes are fixed. OPPO and ORIA show similar speed of increase in network lifetime. When the number of nodes is 100, ORIA achieves twice the network lifetime compared to OPPO. When the number of nodes is 700, ORIA achieves 33 % longer network lifetime. Comparing ORW and ORIA-EDC is the most interesting. ORW and ORIA-EDC significantly outperforms OPPO and ORIA-HOP, by considering the total wakeup time. When the number of nodes is 100, ORIA-EDC achieves much longer network lifetime than ORW. However, as node density increases, the network lifetime of ORIA-EDC grows slower than ORW. So when the number of nodes is 700 and the packet generation rate is 20 packets per 30 s, ORW achieves longer lifetime than ORIA-EDC. This is because when the traffic load is low compared to the number of nodes, the opportunity of in-network aggregation is small. In frequent cases, a node may not receive a single packet during the entire packet hold time. Then the node has wasted energy doing fast wakeups. This result shows that when the traffic load is very low, the cost is larger than the benefit for ORIA-EDC. However, when the traffic load is not too low as in Fig. 7(b), ORIA-EDC achieves longer network lifetime compared to ORW even when the number of nodes is 700.

5.2.2 Impact of traffic load

In the next simulation, packet generation rate was varied to see its impact on performance. Similar to the previous experiment, the normal wakeup time is 2 s, the fast wakeup time is 500 ms, and the packet hold time is 10 s. Figure 8 shows the result. In the figures, the X-axis indicates the number of packets generated in 30 s. The total number of nodes is 200 in Fig. 8(a), and 700 in Fig. 8(b).

As the packet generation rate increases, network lifetime of protocols decrease in general, since more packets need to be transmitted. However, the drop speed of OPPO and ORW is much faster than those of ORIA-HOP and ORIA-EDC. This is because the amount of energy saved from in-network aggregation increases with the traffic load. When half of the nodes generate packets every 30 s, the network lifetime of ORIA-EDC is significantly higher than ORW (300 % at 200 nodes and 150 % at 700 nodes.)

5.2.3 Impact of number of sources

In the previous simulation, the packets are generated at random nodes, which means the packet sources change over time. In this simulation, we choose a subset of nodes as packet sources at the beginning of simulation, and do not change the sources during the run. The packet generation rate of each node is fixed at 1 packet per 30 s. So if 30 nodes are selected as packet sources, the packet generation rate becomes 30 packets per 30 s. The results shown in Fig. 9 has a similar pattern with Fig. 8. When the number of sources is small, ORIA-HOP and ORIA-EDC do not benefit from packet aggregation, and thus have shorter lifetime compared to OPPO and ORW because of the fast wakeup time. As the number of sources increase, ORIA-HOP and ORIA-EDC outperforms OPPO and ORW due to energy savings from packet aggregation.

5.2.4 Impact of packet hold time

The main idea of ORIA is that the nodes hold on to packets while performing fast wakeups in order to promote in-network aggregation, which leads to energy savings. Thus, the length of packet hold time and fast wakeup time will

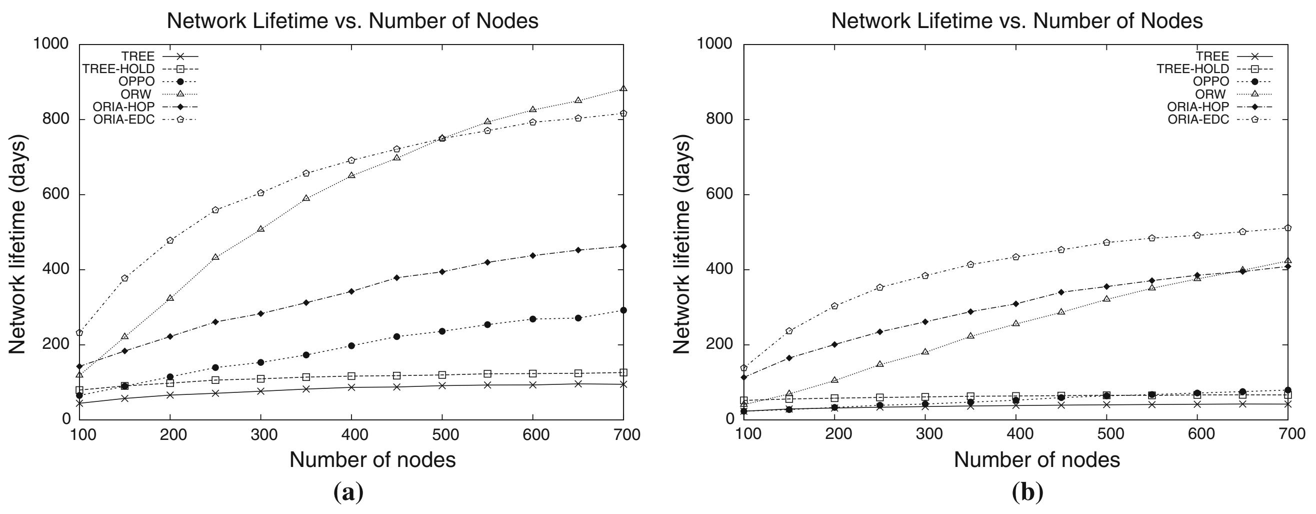

Fig. 7 Network lifetime varying node density. The X-axis indicates the number of nodes, and the Y-axis indicates the time until the first node runs out of energy. **a** 20 packets per 30 s. **b** 100 packets per 30 s

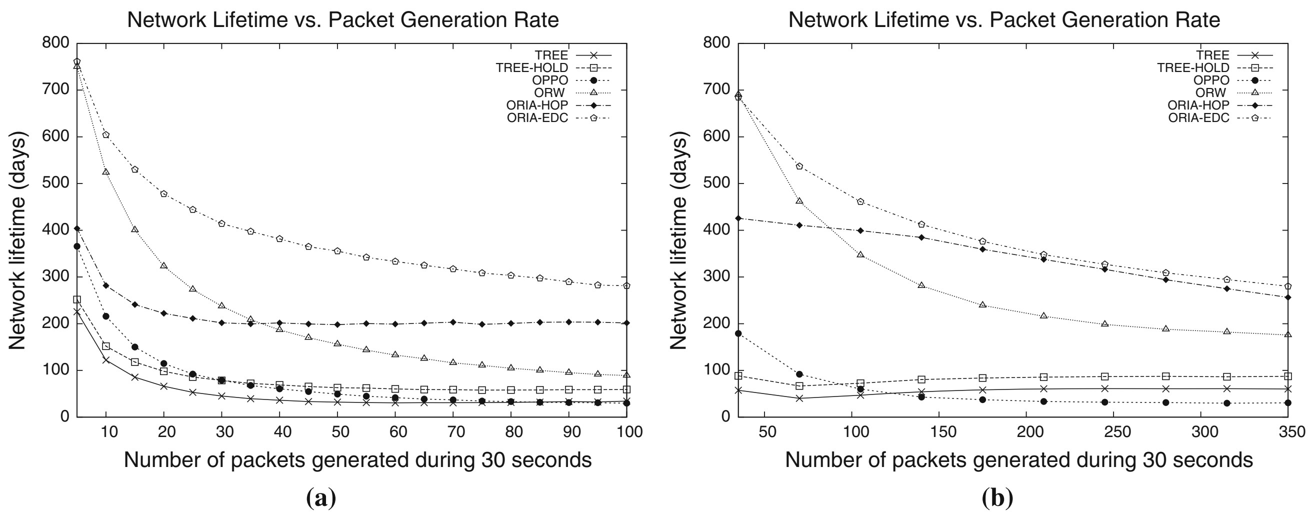

Fig. 8 Network lifetime varying packet generation rate. The X-axis indicates the number of packets generated in 30 s, and the Y-axis indicates the time until the first node runs out of energy. **a** 200 nodes. **b** 700 nodes

have a significant impact on the system performance. The next simulations study the effect of protocol parameters. In Fig. 10, the packet hold time was varied from 0 to 30 s. In Fig. 10(a) and (b), the four lines indicate results when the fast wakeup rate is 500, 1,000, 1,500, and 2,000 ms. In Fig. 10(c) and (d), the four lines indicate results when the fast wakeup rate is 2,000, 3,000, 4,000, and 5,000 ms. The four graphs, Fig. 10(a)–(d), have different node density and packet generation rate in order to study the impact of parameters in various network environments. ORIA-EDC was used for this simulation.

Generally, the network lifetime of ORIA-EDC increases with the packet hold time, since more packets can be aggregated if the packet hold time is long. However, if the fast wakeup time is too short, then a long packet hold time can degrade the performance. There are two reasons for this. First, a node spends more energy when performing fast wakeup. Second, if the nodes wake up too frequently, number of ACK collisions increase which results in increased energy consumption. For example, in Fig. 10(b), when the fast wakeup time is 500ms, the network lifetime decreases when the packet hold time is longer than 15 s. This result shows that even without application delay requirements, long packet hold time may not be always good, if fast wakeup time is not properly set. The optimization of packet hold time and fast wakeup time involves factors such as number of nodes, traffic load and pattern, and normal wakeup interval. Optimizing parameters will be a subject for our future work.

In addition to network lifetime, we show the average packet delay in Fig. 11. The four graphs (a)–(d) have the same number of nodes and packet generation rate as in

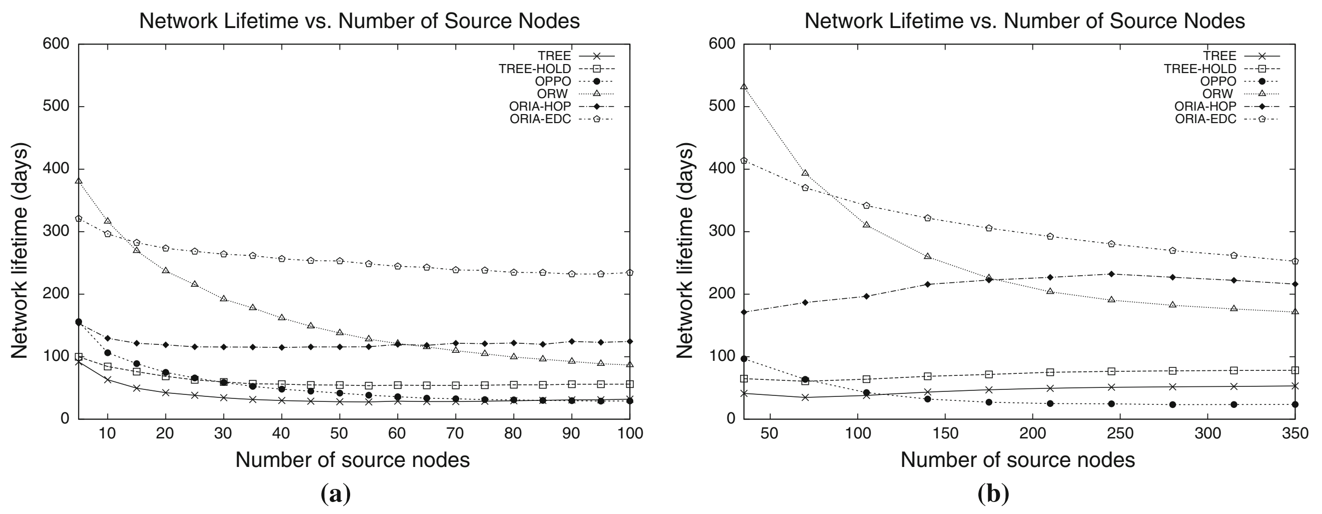

Fig. 9 Network lifetime varying number of sources. The X-axis indicates the number of sources, and the Y-axis indicates the time until the first node runs out of energy. **a** 200 nodes. **b**. 700 nodes

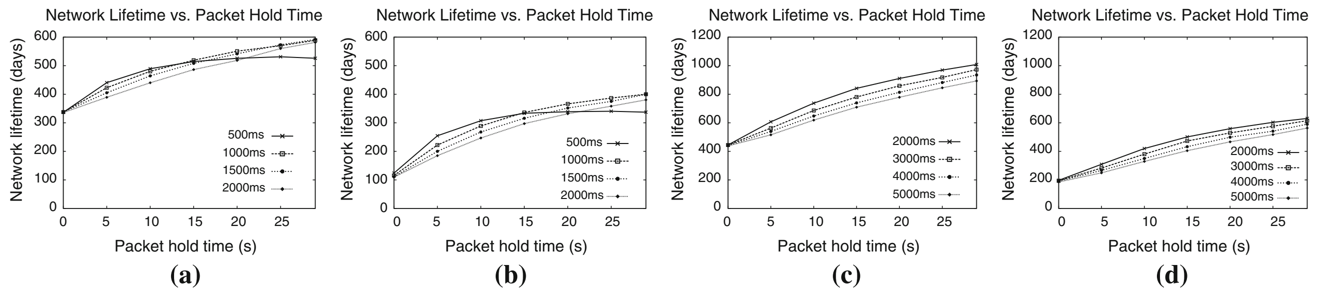

Fig. 10 Network lifetime varying packet hold time. The X-axis indicates the packet hold time and the Y-axis indicates the time until the first node runs out of energy. **a–d** differ in number of nodes (n) and packet generation rate (r). **a** n = 200, r = 20. **b** n = 200, r = 100. **c** n = 600, r = 60 **d**. n = 600, r = 300

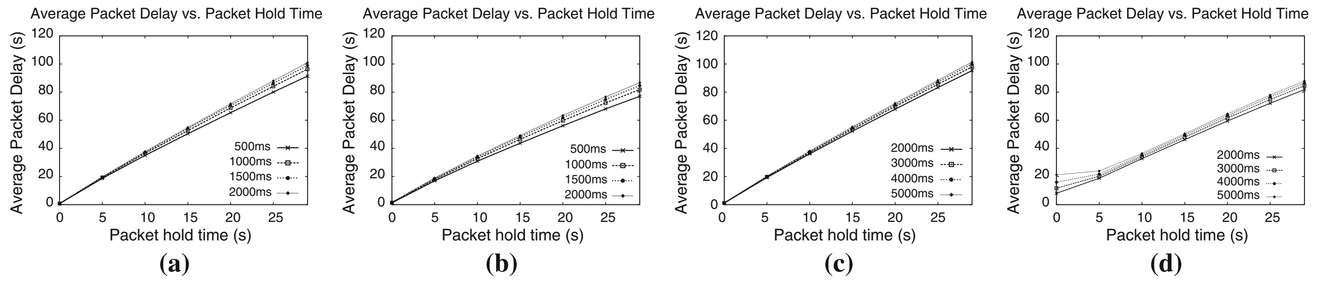

Fig. 11 Average packet delay varying packet hold time. The X-axis indicates the packet hold time and the Y-axis indicates the average time a generated packet reaches the sink. **a–d** differ in number of nodes (n) and packet generation rate (r). **a** n = 200, r = 20. **b** n = 200, r = 100. **c** n = 600, r = 60. **d** n = 600, r = 300

Fig. 10. In general average packet delay increases proportional to the packet hold time, since there is no other factor that significantly affects the packet delay as the packet hold time. When the traffic load is high though, the congestion causes high packet delay, as shown in Fig. 11(d). Even when the packet hold time is 0, the average packet delay is higher than 20 s when the fast wakeup time is 5,000 ms. One possibility regarding packet hold time is that it can be controlled based on the deadline of the packet. Then, packet hold time will be decided based on where the packet is generated. If the packet is generated at a node far away from the sink, the packet hold time of the packet needs to be shorter at each hop compared to packet hold time of a packet generated one hop away from

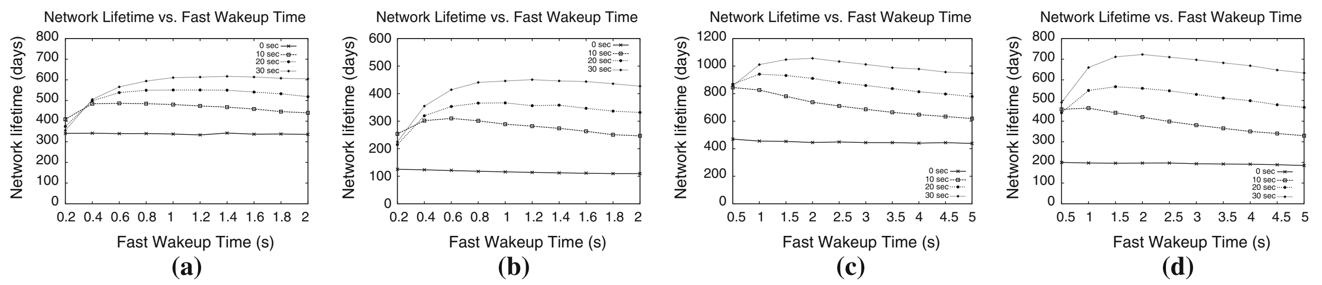

Fig. 12 Network lifetime varying fast wakeup time. The X-axis indicates the fast wakeup time and the Y-axis indicates the time until the first node runs out of energy. **a**–**d** differ in number of nodes (n) and packet generation rate (r). **a** n = 200, r = 20. **b** n = 200, r = 100. **c** n = 600, r = 60. **d** n = 600, r = 300

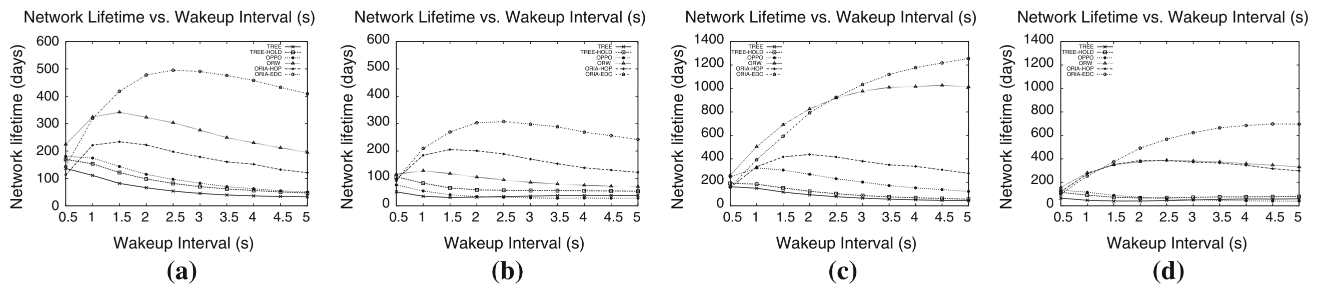

Fig. 13 Network lifetime varying wakeup interval. The X-axis indicates the wakeup interval and the Y-axis indicates the time until the first node runs out of energy. **a**–**d** differ in number of nodes (n) and packet generation rate (r). **a** n = 200, r = 20. **b** n = 200, r = 100. **c** n = 600, r = 60. **d** n = 600, r = 300

the sink. Since multiple packets with different deadlines can be aggregated into a single packet, the packet hold time will be decided based on the earliest deadline.

5.2.5 *Impact of fast wakeup time*

In this simulation, we varied the fast wakeup time from 0.2 to 2 s. Other parameters are the same as the previous simulation. The results are shown in Fig. 11. Similar to the previous simulation, we run simulations on four different environments. For each graph in Fig. 12, the four lines indicate results when the packet hold time is 0, 10, 20, and 30 s.

When the packet hold time is zero, the fast wakeup time has almost no effect, since a node only performs fast wakeup as it waits to acquire the channel and transmit its packet. When the packet hold time is nonzero, the fast wakeup time affects the network lifetime. As the fast wakeup time is shortened (duty-cycle is increased), the network lifetime is improved. However, if the fast wakeup time is too short, the lifetime suddenly degrades, especially when the packet hold time is long. This is due to the reasons previously mentioned. The energy consumption caused from ACK collisions and frequent wake ups degrades the network lifetime when the fast wakeup time is very short. Thus, the fast wakeup time should be set shorter than the normal wakeups, but not too short. In general, setting the fast wakeup time to 30–40 % showed a good performance under various environment.

5.2.6 *Impact of wakeup interval*

This simulation was conducted to study the impact of duty cycle on the protocol performance. In this simulation, we varied the wakeup interval from 500ms to 5000ms. The fast wakeup time is set to be 25 % of the normal wakeup time. The packet hold time is 10 s for TREE-HOLD, ORIA-HOP and ORIA-EDC. The results are shown in Fig. 13. In general, network lifetime of opportunistic protocols (OPPO, ORW, ORIA-HOP, and ORIA-EDC) increases as the wakeup interval becomes larger, but after some point the network lifetime starts to decrease. When the wakeup interval is short, energy consumption from frequent wakeups and ACK collisions degrade the network lifetime. When the wakeup interval is long, the long wait time of senders is the main reason for degradation of network lifetime. When the wakeup interval is very short, ORIA-EDC performs worse than ORW. This is because ORIA-EDC suffers from ACK collisions more than ORW due to fast wakeup. As the wakeup interval increases (duty cycle decreases), the network lifetime of ORIA-EDC overtakes that of ORW. The four graphs show that the optimal wakeup interval depends on the environment, especially node density. The optimal wakeup interval is longer when the network density if higher, because the loss from ACK collision is more severe than the loss from long wait time of senders. This trade-off invites the normal wakeup time to the set of parameters that need optimization, along with packet hold time and fast wakeup time.

6 Conclusion

In this paper we propose an opportunistic routing protocol for data aggregation in wireless sensor networks that works on top of a duty-cycled MAC such as BoX-MAC. The design goal of the proposed protocol ORIA is to increase the chance of in-network aggregation in order to reduce energy consumption and extend network lifetime. A node who possesses a packet to be forwarded waits for other packets to arrive before sending out the packet, and at the same time increases its wakeup rate temporarily so that the possibility of receiving packets from other node becomes higher. In order to avoid negative effects of multiple node reception and ACK collision, the sender sends a CONFIRM packet to the receiver after receiving ACK, thus allowing only a single node to forward its packet. The simulation results confirm that ORIA achieves significantly longer network lifetime compared to a general opportunistic routing protocol and a tree-based protocol, when the traffic load is not too low in the network. This protocol can be used in practice for applications that do not require real-time data collection. For applications with delay requirements, the packet hold time can be controlled in order to minimize energy consumption while meeting the delay requirement. In the future work, we plan to optimize parameters such as packet hold time and fast wakeup rate, based on factors such as number of neighbors, traffic load, link conditions, and residual energy.

Acknowledgments This research was supported by Hallym University Research Fund HRF-2013-015.

References

1. Chipcon, A. S. (2004). Chipcon AS SmartRF CC2420 Preliminary Datasheet (rev 1.2).
2. Biswas, S., & Morris, R. (2005). Exor: Opportunistic multi-hop routing for wireless networks. In Proceedings of ACM SIGCOMM, pp. 133–144.
3. Landsiedel, O., Ghadimi, E., Duquennoy, S., Johansson, M. (2012). Low power, low delay: Opportunistic routing meets duty cycling. In Proceedings of IPSN, pp. 185–196.
4. Krishnamachari, B., Estrin, D., Wicker, S. (2002). Modelling data-centric routing in wireless sensor networks. In Proceedings of IEEE INFOCOM.
5. Bachir, A., Dohler, M., Watteyne, T., Leung, K. (2010). MAC essentials for wireless sensor networks. *IEEE Communications Surveys & Tutorials, 12*(2), 222–248.
6. Akyildiz, I.F., Melodia, T., & Chowdhury, K.R. (2007). A survey on wireless multimedia sensor networks. *Elsevier Computer Networks, 51*, 921–960.
7. Ergen, S. C., & Varaiya, P. (2007). Energy efficient routing with delay guarantee for sensor networks. *Springer Wireless Networks, 13*(5), 679–690.
8. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P. (2009). Collection tree protocol. In Proceedings of the ACM international conference on embedded networked sensor systems, pp. 1–14.
9. Ye, W., Heidemann, J., Estrin, D. (2002). An energy efficient MAC protocol for wireless sensor networks. In Proceedings of IEEE INFOCOM, pp. 1567–1576.
10. van Dam, T., & Langendoen, K. (2003). An adaptive energy-efficient MAC protocol for wireless sensor networks. In Proceedings of ACM Sensys, pp. 171–180.
11. Lin, P., Qiao, C., Wang, X. (2004). Medium access control with a dynamic duty cycle for sensor networks. In Proceedings of WCNC, pp. 1534–1539.
12. Ye, W., Heidemann, J., Estrin, D. (2004). Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *IEEE Transactions on Networking, 12*(3), 493–506.
13. Merlin, C. J., & Heinzelman, W. B. (2010). Duty cycle control for low-power-listening MAC protocols. *IEEE Transactions on Mobile Computing, 9*(11), 1508–1521.
14. Sun, Y., Du, S., Gurewitz, O., & Johnson, D. B. (2008). DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks. In Proceedings of ACM MobiHoc, pp. 53–62.
15. Zhao, Y. Z., Ma, M., Miao, C. Y., Nguyen, T. N. (2010). An energy-efficient and low-latency MAC protocol with adaptive scheduling for multi-hop wireless sensor networks. *Computer Communications, 33*(12), 1452–1461.
16. Zhao, Y. Z., Miao, C. Y., Ma, M. (2012). An energy-efficient self-adaptive duty cycle mac protocol for traffic-dynamic wireless sensor networks. *Wireless Personal Communications, 68*(4), 1287–1315.
17. Polastre, J., Hill, J., Culler, D. (2004). Versatile low power media access for wireless sensor networks. In Proceedings of ACM Sensys, pp. 95–107.
18. Buettner, M., Yee, G., Anderson, E., Han, R. (2006). X-MAC: A short preamble mac protocol for duty-cycled wireless sensor networks. In Proceedings of ACM Sensys, pp. 307–320.
19. Moss, D., & Levis, P. (2008). BoX-MACs: Exploiting physical and link layer boundaries in low-power networking. Stanford University, technical report 08-00.
20. Miller, M., & Vaidya, N. (2005). A MAC protocol to reduce sensor network energy consumption using a wakeup radio. *IEEE Transactions on Mobile Computing, 4*(3), 228–242.
21. Bachir, A., Barthel, D., Heusse, M., Duda, A. (2006). Micro-frame preamble MAC for multi-hop wireless sensor networks. In Proceedings of IEEE ICC, Istanbul, pp. 3365–3370.
22. Park, T., Park, K., Lee, M. J. (2009). Design and analysis of asynchronous wakeup for wireless sensor networks. *IEEE Transactions on Wireless Communications, 8*(11), 5530–5541.
23. Aonishi, T., Matsuda, T., Mikami, S., Kawaguchi, H., Ohta, C., Yoshimoto, M. (2006). Impact of aggregation efficiency on GIT routing for wireless sensor networks. In Proceedings of IEEE international conference on parallel processing workshops, pp. 151–158.
24. Mottola, L., & Picco, G. (2010). Muster: Adaptive energy-aware multi-sink routing in wireless sensor networks. *IEEE Transactions on Mobile Computing, 10*(12), 1694–1709.
25. Zorzi M., & Rao, R. (2003). Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Multi-hop performance. IEEE Transactions on mobile computing, pp. 337–348.

26. Gu Y., & He T. (2007). Data forwarding in extremely low duty-cycle sensor networks with unreliable links. In Proceedings of ACM Sensys, pp. 321–334.
27. Liu, C. & Cao, G. (2010). Distributed monitoring and aggregation in wireless sensor networks. In Proceedings of IEEE INFOCOM, pp. 1–9.
28. Villas, L., Boukerche, A., Ramos, H., de Oliveira, H., de Araujo, R., Loureiro, A. (2013). DRINA: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *IEEE Transactions on Computers, 62*(4), 676–689.
29. Aitsaadi, N., Blaszczyszyn, B., Muhlethaler, P. (2012). Performance of opportunistic routing in low duty-cycle wireless sensor networks. In Proceedings of IFIP wireless days, pp. 1–3.
30. Jurdak, R., Ruzzelli, A., O'Hare, G. (2008). Adaptive radio modes in sensor networks: How deep to sleep? In Proceedings of IEEE SECON, pp. 386–394.

Author Biographies

Jungmin So received the B.S. degree in computer engineering from Seoul National University in 2001, and Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign in 2006. He is currently an assistant professor in Department of Computer Engineering, Hallym University. His research interests include wireless networking and mobile computing.

Heejung Byun received the B.S degree from Soongsil University, Korea, in 1999, the M.S. degree from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 2001, and the Ph.D. degree from KAIST in 2005. She was a senior researcher in Samsung Electronics, Ltd. from 2007 to 2010. She is currently a professor with the Department of Information and Telecommunications Engineering, Suwon University, Korea. Her research interests include network protocol, network modeling, controller design, and performance analysis.