EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Opportunistic routing with in-network aggregation for duty-cycled WSNs with delay requirements

Jungmin So[1] and Heejung Byun[2*]

Abstract

This paper proposes an opportunistic routing protocol for wireless sensor networks that works on top of an asynchronous duty-cycling medium access control (MAC) protocol. The proposed protocol is designed for applications that are not real-time but still have some requirements on packet delay. The main idea is that if a packet has time to spare, it can wait on a node hoping that it can be aggregated with other packets, resulting in reduced number of transmissions. The forwarders and the packet hold time depend on the energy status of nodes in the network. The simulation results show that the proposed protocol achieves longer network lifetime compared to the other state-of-the-art protocols, while satisfying application delay requirements.

Keywords: Wireless sensor networks; Opportunistic routing; In-network aggregation; Delay requirements

1 Introduction

It is well known that energy efficiency is the most important factor in designing protocols for wireless sensor networks, since the network may need to operate without human intervention for as long as possible. Medium access control and routing protocols are designed to minimize energy consumption while achieving application requirements. In order to conserve energy, it is essential to put nodes into sleep mode as much as possible. For this reason, many medium access control protocols run duty-cycles in which nodes periodically switch between active and sleep modes.

Medium access protocols running duty-cycles can be divided into synchronized and asynchronous protocols. In synchronized protocols, some nodes (or all nodes) synchronize their active periods so that they wake up at the same time and communicate if necessary. However, maintaining active period synchronization can be expensive, because it requires periodic message exchange between nodes. Also, protocols requiring tight synchronization may not work well in practice due to the slight differences in clock speed of the nodes. Asynchronous protocols do not synchronize active periods, so nodes do not wake up at the same time. Thus, when a node wants to send a packet to its neighbor, it is probable that the intended receiver is not awake. To communicate, either sender or receiver should wait in the active mode for its counterpart to wake up. Since there is no requirement for time synchronization, asynchronous protocols are easier to implement and operate than synchronized protocols. B-MAC [1], X-MAC [2], and BoX-MAC [3] are widely known asynchronous protocols, and BoX-MAC is selected as the basic medium access control (MAC) protocol for TinyOS 2.x [4].

With BoX-MAC, the major source of energy consumption is the time sender waits for receiver to wake up. For a tree-style routing protocol such as CTP [5], a sender has to wait in active mode for half of the wake-up interval, in average. On the other hand, opportunistic routing protocols such as Opportunistic Routing in Wireless sensor network (ORW) [6] work particularly well with asynchronous MAC protocols. Instead of selecting a particular forwarder, each node maintains multiple candidate forwarders and forwards its packet to one of the candidates who wakes up first. Opportunistic routing reduces packet delay, but more importantly, it can save significant amount of energy by reducing the wait time of senders.

*Correspondence: heejungbyun@suwon.ac.kr
[2]Department of Information and Telecommunications Engineering, Suwon University, San 2-2 Wau-ri, Bongdam-eup, Hwaseong 445-743, South Korea
Full list of author information is available at the end of the article

In wireless sensor network routing, in-network aggregation can also play a significant role in reducing energy consumption. If multiple packets can be combined at an intermediate node, the number of transmissions reduces, even though the packet size can be increased depending on how the data are combined. Since there is a fixed overhead of sender wait time for each packet transmission, reducing the number of transmissions is also important in extending network lifetime. For in-network aggregation, a node has to hold on to packets for some time before transmitting. Meanwhile, if other packets arrive at the node, data can be aggregated and sent as a single packet. If the packets are generated at random times, there is a trade-off between packet delay and energy consumption: if nodes hold on to the packet longer, more packets will likely to be aggregated, but packet delay will increase. This trade-off means that the energy consumption can be controlled based on application delay requirements.

This paper proposes a routing protocol that controls this trade-off and achieves better network lifetime compared to other tree-style or opportunistic protocols. The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes preliminary backgrounds and presents the proposed protocol in detail. Section 4 shows the performance of protocols using simulations. Finally, Section 5 concludes with remarks on future work.

2 Related work

A lot of medium access control and routing protocols have been developed for wireless sensor networks (WSNs) in the last decade [7,8,33-35]. Although WSNs share similarities with mobile *ad hoc* networks and wireless mesh networks, protocols for WSNs are unique in that they are focused on achieving energy efficiency rather than high throughput. Most WSN applications do not require high throughput, although there are exceptions such as multimedia sensor networks [9]. Some applications have quality of service requirements such as reliability [10] and packet delay [11].

Most medium access control protocols use duty-cycling for energy efficiency, and the protocols can be divided into two categories: synchronized and asynchronous. In synchronized protocols, all nodes or a subset of nodes synchronize their active times, so that they can wake up at the same time. In SMAC [12], all nodes wake up at the same time and stay in active mode for some time before going back to sleep. Time synchronization is maintained using SYNC messages exchanged during active period. TMAC [13] and DSMAC [14] are improvements to the SMAC, where active periods are adaptively managed to conserve energy. The most difficult problem in operating synchronized MAC protocols is to maintain tight time synchronization among nodes. Loss of synchronization can lead to continuous packet drops causing significant waste of energy.

Asynchronous protocols do not synchronize active periods, so nodes wake up at different times. If a node wants to transmit a packet, it has to wake up and stay in active mode until the receiver wakes up. B-MAC [1], X-MAC [2], and BoX-MAC [3] are well-known asynchronous protocols. In B-MAC, the transmitter wakes up and transmits a preamble for the duration of a whole wake-up interval (assuming all nodes have the same wake-up interval). If a node wakes up and detects a preamble, it stays in active mode. After transmitting the preamble, the transmitter transmits the packet. All neighbors of the transmitter will be awake and receive the packet by then. The problem with B-MAC is that even for a unicast packet, the sender wakes up all its neighbors unnecessarily. In X-MAC [2], the sender sends short preambles repeatedly, instead of a long preamble. The short preamble indicates who the receiver is. When a neighbor node wakes up, it checks the preamble to see if it is the intended receiver. If not, the node goes back to sleep. If yes, the node sends an ACK to the sender. On receiving ACK, the sender sends data to the receiver. By doing this, nodes that are not the receiver can avoid waiting for the whole wake-up interval in active mode. In BoX-MAC [3], the sender sends the packet itself in streams instead of short preambles. When the receiver receives the packet, it sends back an ACK and both can go to sleep without going through another transaction. The most significant benefit of using an asynchronous protocol is that it is simple to implement and easy to operate.

A routing protocol, which decides paths of the packets, sits between the MAC layer and the application layer. Similar to the MAC protocol, the design goal of a routing protocol is to minimize energy consumption while achieving application requirements. The basic strategy of routing is to build a tree rooted at the sink, so that the packets are forwarded along the tree branches [5]. The tree can be constructed based on various factors such as number of hops, residual energy, and expected transmission count (ETX) [15]. Opportunistic routing is another strategy of forwarding packets. In ExOR [16], an opportunistic routing protocol for *ad hoc* networks, a node chooses a list of candidate forwarders instead of just one forwarder as in a tree-based protocol. Each forwarder is given a priority, which is included in the packet header. A node receiving the packet waits for a delay that is computed based on its priority before transmitting the packet. Other nodes that overhear this packet cancel their transmissions. Although it is shown that ExOR achieves longer network lifetime compared to a tree-based protocol in an *ad hoc* network, it cannot be used in a duty-cycled WSN since not all nodes are awake when the sender transmits its packet. GeRaF [17] is a similar protocol,

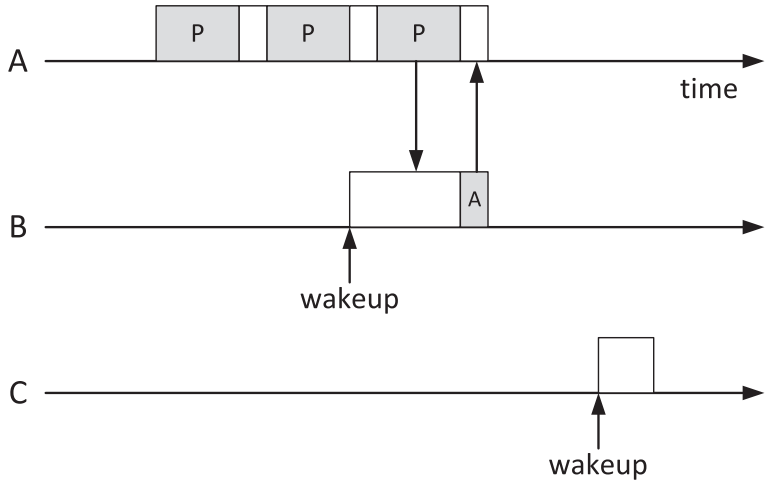

Figure 1 Behavior of ORW running on top of asynchronous duty-cycled MAC. Node B and C are in node A's forwarder set. When node A is sending packets, node B wakes up before node C and receives the packet. Node B replies by sending an ACK to node A, who stops sending packets.

but forwarders are selected based on location information. DSF [18] is an opportunistic protocol that works with duty cycling. A node knows the sleep schedule of the neighbors and selects forwarders based on the schedule, as well as delay, reliability, and energy status. ORW [6] is a routing protocol specifically designed to work with an asynchronous duty-cycled MAC protocol. A node chooses a set of candidate forwarders and forwards its packet to whoever wakes up first, after the node starts transmitting packets. This strategy works particularly well with BoX-MAC, since the sender sends the packet repeatedly until a receiver receives the packet and replies by sending an ACK. The forwarder set is determined based on a metric called expected duty cycled (EDC wake-ups), which takes into account the number of neighbors a node has.

There are routing protocols designed to benefit from in-network aggregation [19]. The greedy incremental tree (GIT) [20] is a heuristic strategy of building an aggregation tree so that the number of packet transmission is minimized. Aonishi et al. [21] improved GIT by considering the aggregation efficiency. Liu et al. [22] considered an environment where there is an upper limit to the number of packets that can be aggregated. DRINA [23] is a cluster-based routing protocol for event-triggered packet aggregation. When an event occurs, multiple nodes located nearby detect the event and generate packets. The nodes form a cluster and elect a leader which becomes the point of aggregation. The packets are first gathered at the cluster head and then sent to the sink as an aggregated packet.

There are many other approaches for achieving energy efficiency in wireless sensor networks. EDAL [24] is a data collection protocol that considers deadlines for packet delivery. To meet the deadlines while achieving energy efficiency, EDAL leverages results from open vehicle routing problem and provides heuristics that can be applied to WSNs. Cross-layer design of routing and MAC protocol can be beneficial to preserving energy as shown in [25], where DSR was modified to provide energy-efficient routing in sensor networks. Compressed sensing, which is a signal processing technique which delivers a full recovery of signals from far fewer measurements, has been applied for energy-efficient data collection in

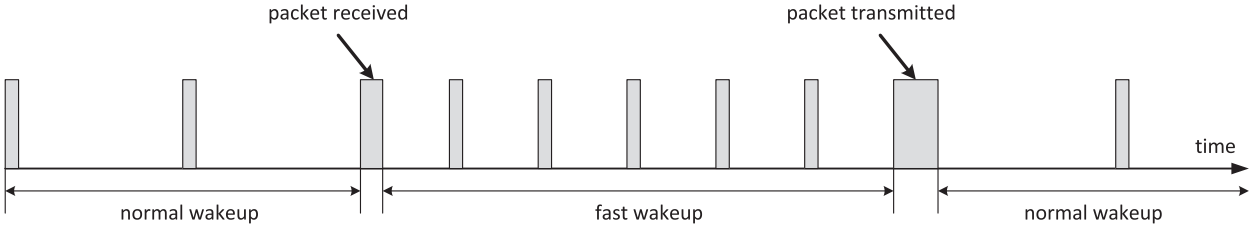

Figure 2 The wake-up behavior of ORIA.

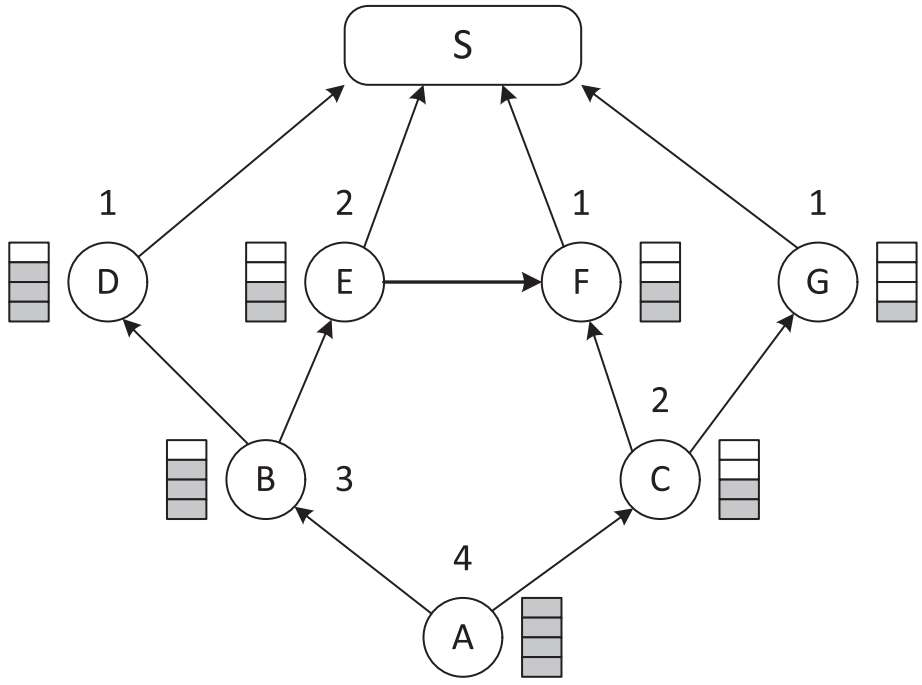

Figure 3 An example scenario. The arrows indicate the forwarder set. For example, nodes B and C are in the forwarder set of node A. The bars next to the nodes indicate residual energy. The number indicates the maximum number of hops to the sink.

sensor networks [26]. Prediction can be useful for WSNs due to the temporally redundant nature of sensor data, and CoGKDA [27] uses grey model and Kalman filter to collect data with high prediction accuracy and low communication overhead. Finally, Compressive Data Collection (CDC) [28] observes that real data sets from sensor networks may have strong compressibility and the power-law decaying data model fits well for these data. Thus, CDC uses on-the-fly compression of sensor data in order to reduce communication overhead and energy consumption.

Opportunistic Routing with In-network Aggregation (ORIA) [29] is an opportunistic routing protocol similar to ORW but tries to benefit from in-network aggregation. To promote aggregation, each node holds on to its packet for a certain duration before transmitting, and while the node is holding a packet, it increases its wake-up rate in order to increase the chance of receiving packets from downstream nodes. ORIA achieves longer network time compared to ORW, but since ORIA uses fixed packet holding time at each node, the packet delay from source to sink may become long. If there is a packet delay requirement, the routing protocol should deliver the packets in time. The proposed protocol described in the next section considers packet delay requirements, while pursuing the benefit of opportunistic routing and in-network aggregation. It is what makes the proposed protocol different from other existing protocols.

3 Proposed protocol

3.1 Preliminaries: ORW and ORIA

ORW is an opportunsic protocol designed to work on top of BoX-MAC, an asynchronous duty-cycled MAC protocol. In ORW, each node maintains a forwarder set, which is a list of candidate forwarders selected among its neighboring nodes. When a node has a packet to transmit,

Table 1 Default parameters used for simulation

Parameter	Value
Simulation area	100 × 100 m
Transmission range	20 m
Carrier sense range	40 m
Backoff time when carrier is busy	30 ms
Wake-up interval	1 s
Short wake-up interval	0.5 s
Delay requirement	30 s
Initial energy	2,000 mAh
Energy consumption Tx	17.4 mA [30]
Energy consumption Rx	19.7 mA [30]
Energy consumption for mode switching	0.3 mA [31]
Active listening time when channel is idle	5.61 ms [3]
Active listening time when channel is busy	20 ms [3]
Packet transmission/receive time	50 ms [3]

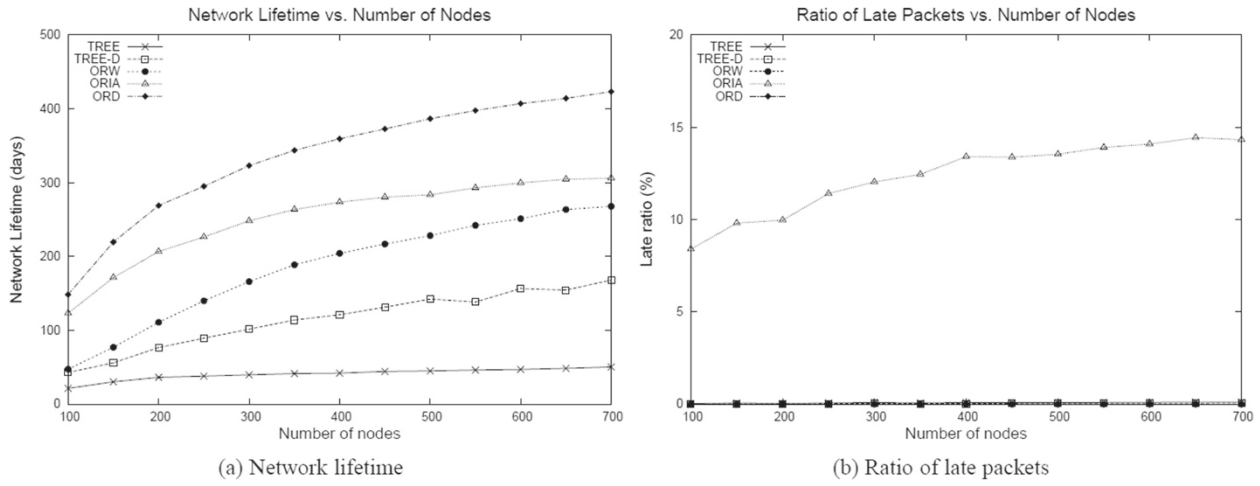

Figure 4 Results varying number of nodes in the network. Packet generation rate is three packets per second.

it wakes up, senses the channel, and starts transmitting packets repeatedly if the channel is idle. The forwarder set is included in the packet header. If a neighboring node which is in the forwarder set of the sender wakes up and receives the packet, it sends an ACK back to the sender and the sender stops transmitting packets. Figure 1 shows the behavior of ORW. In the figure, node A is the sender and nodes B and C are in A's forwarder set. When node A generates a packet, it wakes up and starts transmitting packets. Suppose node B wakes up earlier than node C. Node B wakes up, receives the packet, and sends back an ACK. On receiving ACK, node A stops transmission and goes back to sleep. When node C wakes up, node A is no longer transmitting.

The forwarder set is decided based on EDC, which is calculated as follows. For a subset S_i of node i, EDC_i is:

$$\mathrm{EDC}_i(S_i) = \frac{1}{|S_i|} + \frac{\sum_{j \in S_i} \mathrm{EDC}_j}{|S_i|}. \quad (1)$$

In the equation, the first term is the expected wait time of the node, which is inversely proportional to the number of neighbors. The second term is the average expected wait time after the packet is forwarded to the next hop. The sum of these two terms is the expected wait time experienced from the node to the sink. It is shown in [6] that using EDC to select forwarders achieves longer network lifetime compared to just using number of hops.

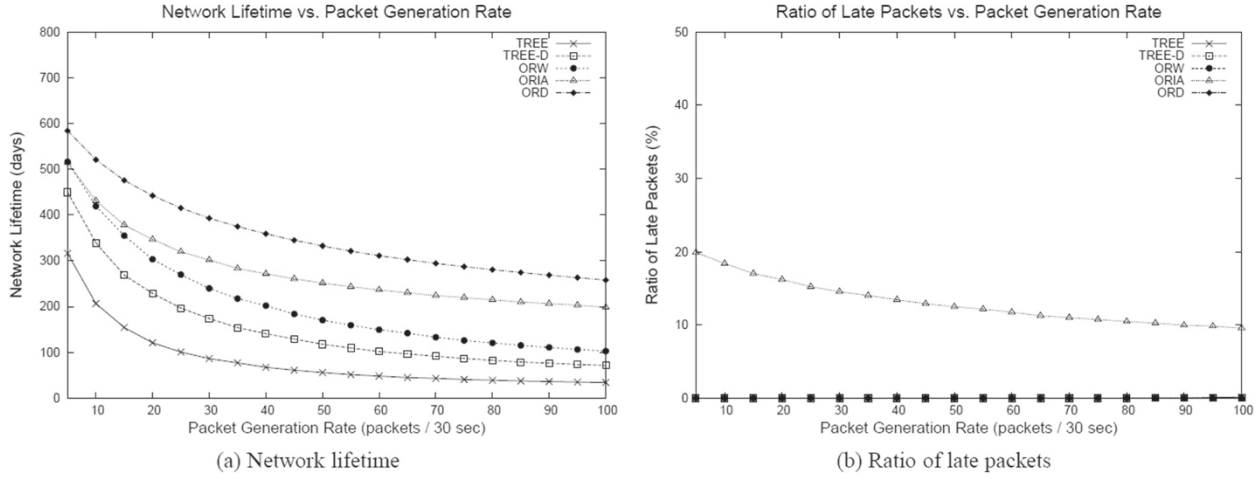

Figure 5 Results varying traffic load. The number of nodes is 200.

ORIA is an extension to ORW, which tries to benefit from in-network aggregation. In the scenario shown in Figure 1, suppose node C is holding a packet to send. Then, it might be better for node A to send its packet to C rather than B. The energy consumption of node A might slightly increase due to the longer wait time, but the upstream nodes save energy because two packets are aggregated at node C. So, the idea of ORIA is simple: let each node hold on to the packet for some time, and if a node is holding a packet, increase its wake-up rate so that it can receive packets from downstream nodes with higher probability. In ORIA, the hold time is fixed. For example, if the hold time is 5 s, a node holds on to the packet for 5 s after receiving the packet or generating the packet, before sending it to its upstream node. The hold time only applies to the first packet received or generated, which means the hold time is not extended even if a new packet arrives during the hold time. Figure 2 shows the wake-up behavior of nodes running ORIA.

The evaluation of ORIA shows that ORIA achieves longer lifetime compared to ORW, if long packet delay can be tolerated by the application. In ORIA, packets stay at each node for a fixed delay (e.g., 10 s), unless the packet is aggregated with another packet on reception (in the case which the hold time is based on the previous packet). Thus, a packet generated at a node six hops away from the sink can take 1 min until the sink receives the packet. However, for a lot of applications, the value of information decreases as the delivery time is increased. Even for applications that are not time-critical, it is useful if packets are delivered to the sink based on delay requirements given by the application. The proposed protocol, called ORD (Opportunistic Routing with Delay requirements), provides a tuning knob so that the trade off between energy consumption and delay requirement can be controlled.

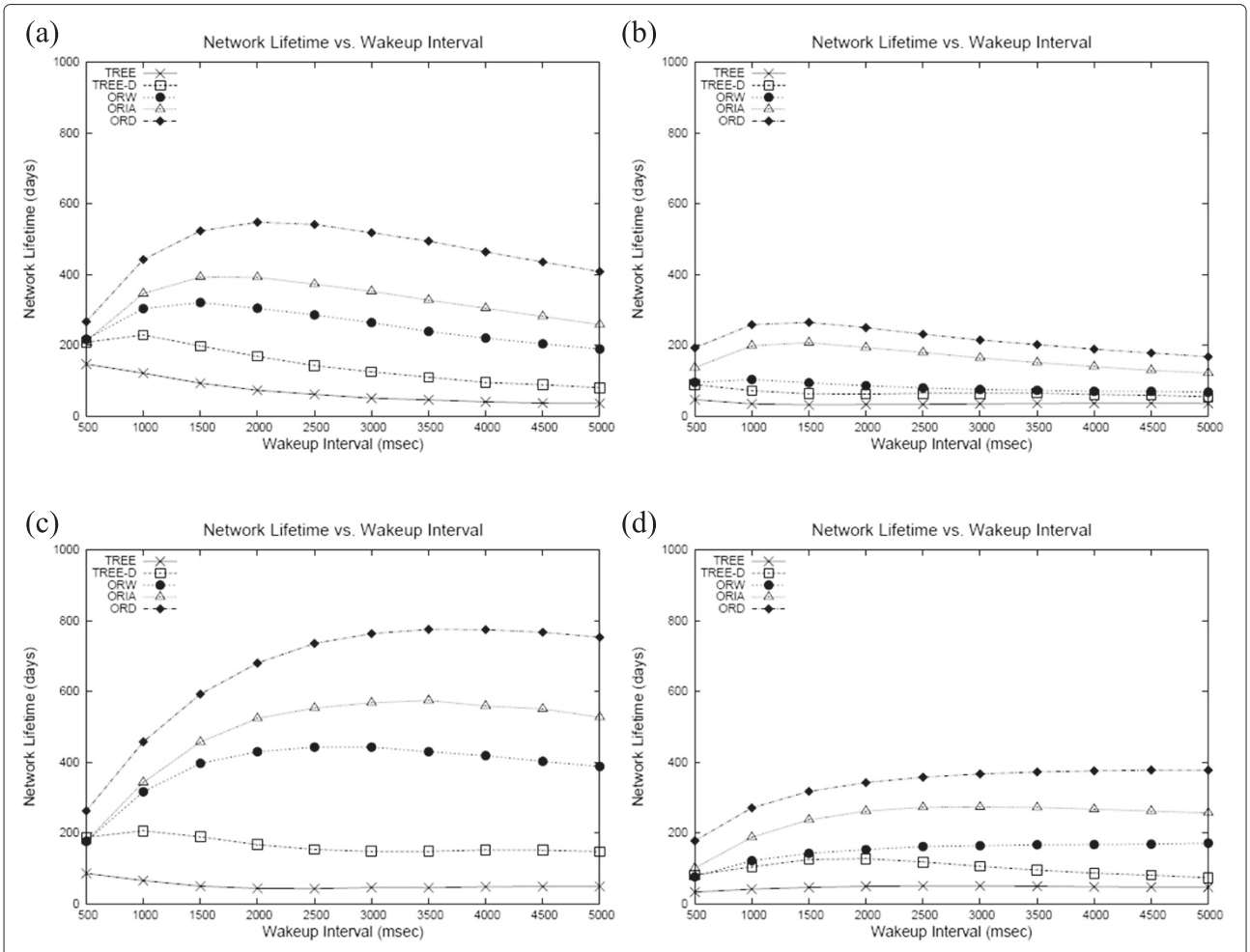

Figure 6 Network lifetime varying wake-up interval. In the graphs, *n* is the number of nodes and *s* is the number of packets generated per 30 seconds. **(a)** $n = 200$, $s = 20$. **(b)** $n = 200$, $s = 100$. **(c)** $n = 600$, $s = 60$. **(d)** $n = 600$, $s = 300$.

3.2 Proposed protocol: ORD

3.2.1 Initial operation

The initial operation of ORD is similar to that of ORW and ORIA. When a node is deployed, it starts duty-cycling by switching between active and sleep modes. Also, it broadcasts a HELLO-REQ message to its one-hop neighbors. On receiving the HELLO-REQ message, the neighbors send HELLO messages to the new node. The HELLO message includes ID, Expected Duty Cycled (EDC) wakeups, and hop distance from the sink. After receiving HELLO messages from the neighbors, the node computes its EDC using Equation 1 and also computes the forwarder set. Once this is done, the node broadcasts a HELLO message so that its neighbors can update its EDC and forwarder set. Note that HELLO messages sent in response to HELLO-REQ are unicast packets, whereas other HELLO messages are broadcast.

Once a node computes its forwarder set, it is ready to generate packets or forward packets from other nodes. To adapt to possible topology changes, each node periodically broadcasts a HELLO message to its one-hop neighbors.

3.2.2 Status monitoring and feedback

In ORD, each node maintains the following information for each forwarder.

- r: residual energy of each forwarder
- m: maximum number of hops to the sink to each forwarder

Consider the scenario in Figure 3. Note that the arrows in the figure indicate the forwarder sets, not the links. For example, B and C are in node A's forwarder set, D and E are in B's forwarder set, and so forth. The bars next to the nodes indicate the residual energy, denoted by r. For example, r_A is 100, while r_B is 75, and r_C is 50. The granularity of expressing residual energy needs not be too fine, and 4 bits (16 levels) are used in the simulations. The

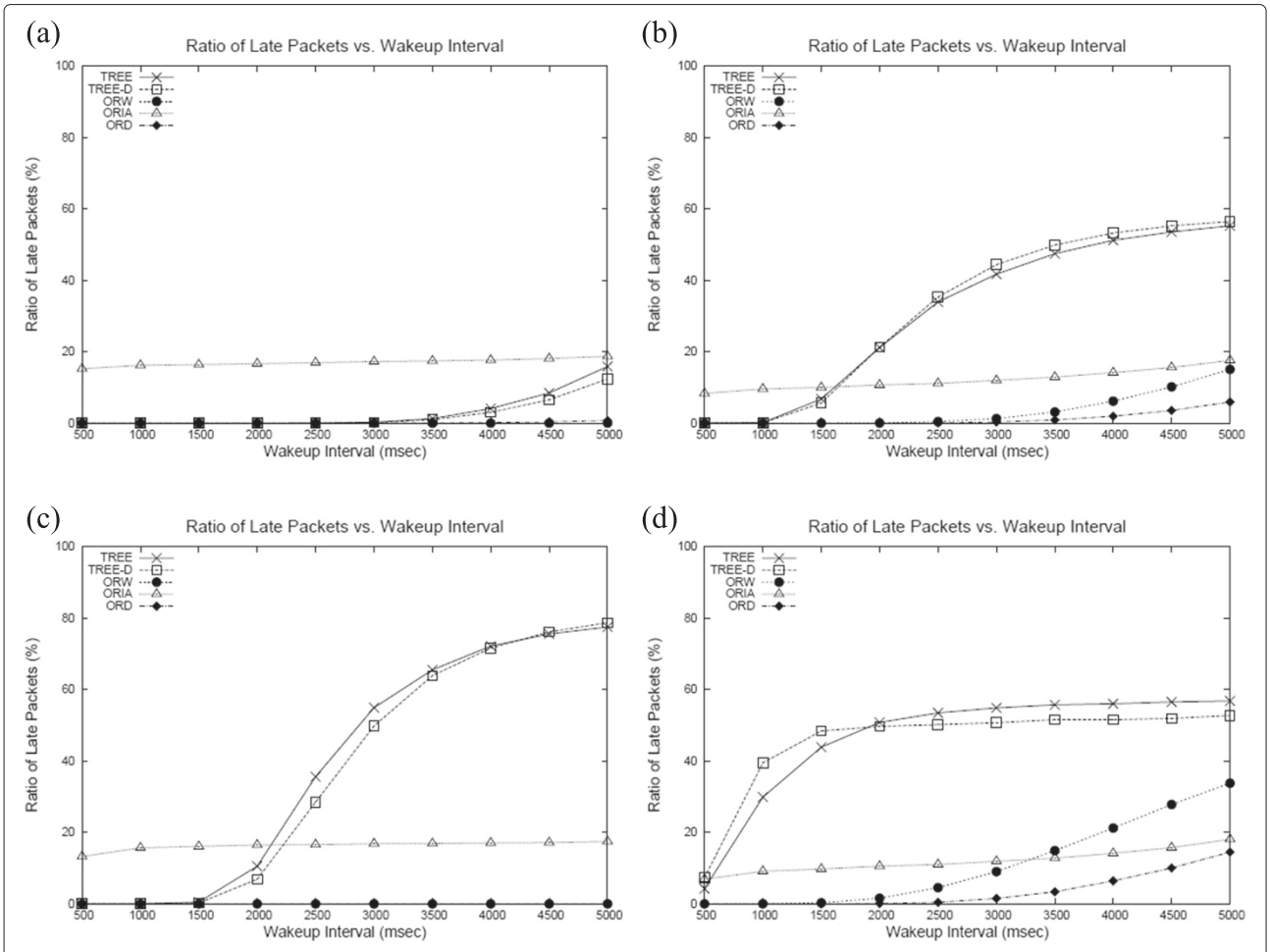

Figure 7 Ratio of late packets varying wake-up interval. In the graphs, n is the number of nodes and s is the number of packets generated per 30 seconds. **(a)** $n = 200$, $s = 20$. **(b)** $n = 200$, $s = 100$. **(c)** $n = 600$, $s = 60$. **(d)** $n = 600$, $s = 300$.

number next to the node is the maximum number of hops to the sink. Since the forwarders are selected based on EDC, it is possible that a node with the same hop distance to the sink in a shortest path tree can become a forwarder in ORD. For example, node F is in the forwarder set of node E. Because of this, although the shortest hop distance from node A to the sink is three hops, the packet can actually take four hops to reach the sink. Thus, m_E is 2, m_B is 3, and m_A is 4.

When a node receives a packet, it sends back an ACK to the sender. The values r and m are included in the ACK so that the downstream node can update its value. As described next, forwarder sets can be changed based on residual energy in ORD, although the changes do not occur very dynamically. Thus, nodes need to keep track of updated values in order to make proper decisions.

3.3 Forwarder selection and packet hold time

Consider the scenario in Figure 3 again. Suppose node B has to forward a packet. Node B has nodes D and E in its forwarder set, but the energy level of node E is lower than node B. If the energy level of the forwarder is lower by a certain threshold than the sender, the sender excludes the node from the forwarder set temporarily. If all nodes are excluded from the forwarder set, then the nodes with maximum energy level are included in the forwarder set. Node E is excluded from B's forwarder set, so B forwards its packet to only node D. Once node D's energy level is brought down to 50%, then B's forwarder set includes nodes D and E again, since both nodes are at the same energy level and that is the maximum energy level among nodes in B's forwarder set. In the figure, node A's forwarder set will only include node B, since it has larger amount of energy left than node C. Suppose node A's energy level was 25%. In this case, A's forwarder set will include both B and C.

The rationale behind this is that if the sender's energy level is low, the first priority is to reduce the sender wait time. On the other hand, if the sender has enough energy, then it is better to forward the packet to the forwarder

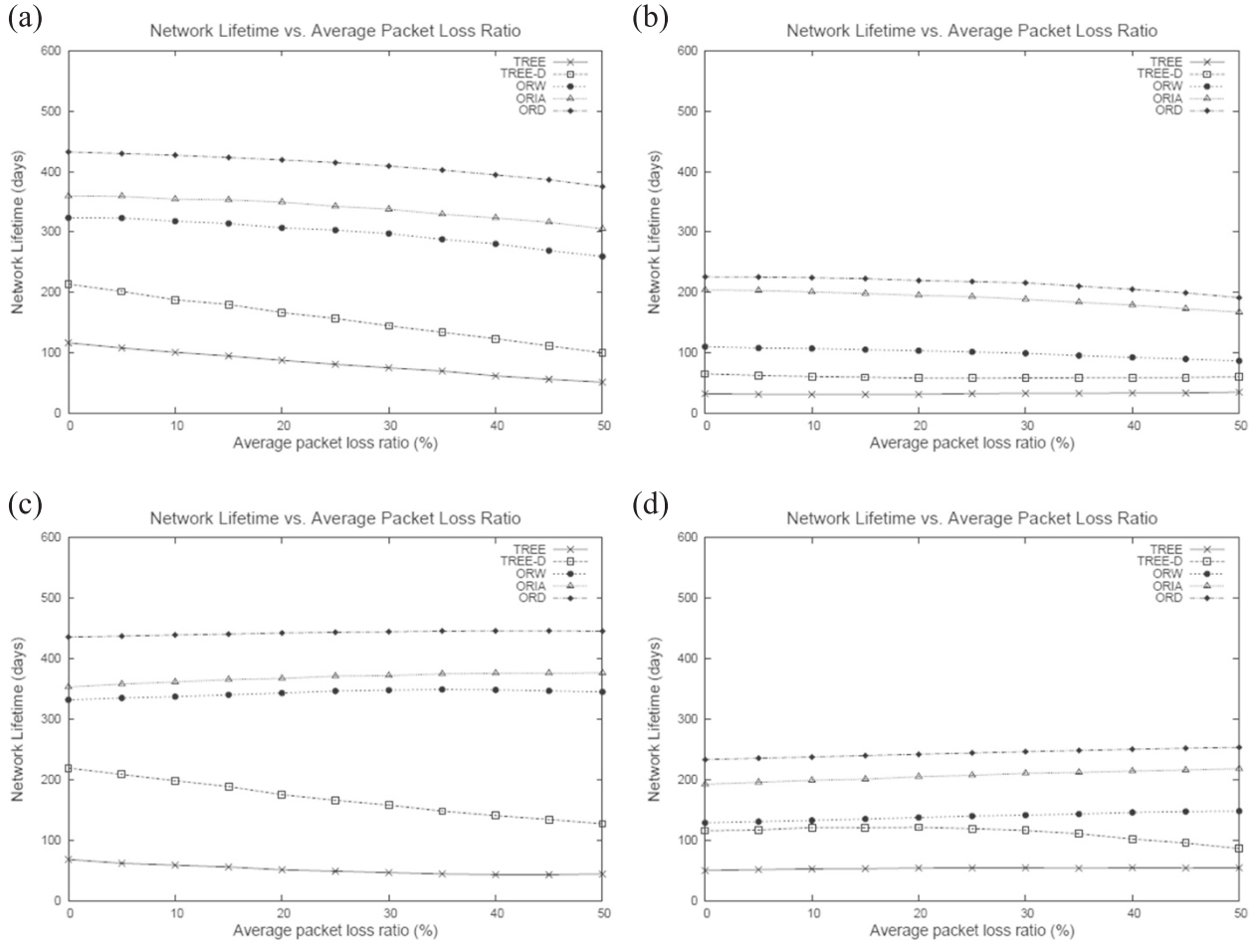

Figure 8 Network lifetime varying average packet loss ratio. In the graphs, n is the number of nodes and s is the number of packets generated per 30 seconds. **(a)** $n = 200$, $s = 20$. **(b)** $n = 200$, $s = 100$. **(c)** $n = 600$, $s = 60$. **(d)** $n = 600$, $s = 300$.

with higher energy level in order to balance the energy level of the nodes. The forwarder selection algorithm is described as a pseudocode in Algorithm 1.

Algorithm 1 Pseudocode for forwarder selection algorithm.

Algorithm for selecting forwarders:

$F_i' \leftarrow \emptyset$, $r_{\max} \leftarrow 0$
for each node k in the forwarder set F_i **do**
  if $r_k \geq r_i$ **then**
    $F_i' \leftarrow F_i' \cup \{k\}$
  end if
  if $r_k > r_{\max}$ **then**
    $r_{\max} \leftarrow r_k$
  end if
end for
if $F_i' = \emptyset$ **then**
  for each node k in F_i **do**
    if $r_k = r_{\max}$ **then**
      $F_i' \leftarrow F_i' \cup \{k\}$
    end if
  end for
end if

Once a node selects its forwarders, it has to decide how much it should hold the packet before transmitting. The hold time in ORD is based on m, which is the maximum number of hops to the sink. A node keeps track of m for each forwarder, so it can compute its own m value as follows. For node i, m_i is:

$$m_i = \max m_k + 1, k \in F'. \tag{2}$$

Now, the packet hold time is calculated as follows.

$$t_h = \frac{D_{\text{req}} - t_e}{m_i \times t_w} - g \tag{3}$$

where t_h is the packet hold time, t_e is the elapsed time from when the packet is generated, t_w is the wake-up interval, and g is a margin which considers time used for channel contention and backoff. The packet generation time is stamped in the packet header. Similar to ORIA, the hold time is applied only to the first packet received or generated at the node.

Consider the scenario in Figure 3. Suppose delay requirement is 30 s, and delays such as channel contention delay, transmission time, and propagation time are ignored. When node A generates a packet, it holds the packet for 7.5 s since m_A is 4. Now, the packet may be forwarded to B or C. If the packet goes to B, B holds the packet for 7.5 s, since m_B is 3 and 22.5 s are left before the time expires. However, if the packet happens to go to node C, node C holds the packet for 11.25 s, since m_C is 2. In this way, packets are generally held for longer duration of time as the packet gets nearer to the sink. This is beneficial to maintaining the network alive, since nodes closer to the sink tend to consume more energy.

4 Performance evaluation

4.1 Simulator and setup

Performance evaluation is conducted using a simulator written in C++. The simulator is event-driven and models real-life events such as carrier sense, backoff, and packet collisions. When the simulation starts, nodes are randomly placed in the simulation area. The sink node is placed at the center. Once deployed, nodes exchange messages with each other to gain information necessary for routing. (e.g., tree-based protocols need to know the hop distance of each node from the sink, opportunistic protocols need to compute EDC and forwarder sets). The overhead of exchanging messages at this stage is not included in the total energy consumption, since it is assumed to be done once at the deployment stage, or rarely done during operation.

After the initial phase is over, nodes begin their duty cycles at random time, waking up and going to sleep asynchronously. Also, according to the given packet generation rate, nodes randomly generate packets. If a node generates a packet, it is forwarded to the sink using the routing protocol. If multiple packets meet at an intermediate node, the packets are aggregated. Perfect aggregation is assumed here, meaning packet size stays the same even when multiple packets are aggregated. When a node wants to transmit a packet, it wakes up and senses the channel. If the channel is busy, the node goes back to sleep and tries again after backoff time. The sink node is assumed to be always active. The simulation continues until one of the nodes drains all of its energy and dies.

Performance of ORD is compared against the following protocols.

1. TREE: A tree-based protocol. Each node selects a parent based on the hop distance from the sink. A node forwards its packet to its parent.
2. TREE-D: A tree-based protocol with dynamic parent selection. In this protocol, nodes periodically reselect their parents. A neighbor who is closer to the sink and has the largest amount of residual energy is selected as the parent. This reselection leads to load balancing and thus extension of network lifetime.
3. ORW: This is the protocol described in [6]. Each node calculates its EDC and forwarder set. The packet is forwarded to one of the node in the forwarder set that wakes up first and receives the packet.

4. ORIA: This protocol, proposed in [29], builds upon ORW to promote in-network aggregation. When a node generates or receives a packet, it holds the packet for a while before transmitting the packet. Meanwhile, the node wakes up more frequently so that it can increase possibility of receiving packets from its neighbors.

The default parameters used in the simulations are listed in Table 1, unless otherwise specified. Most of the parameters including simulation area size, transmission range, backoff time, wake-up interval, and short wake-up interval are selected to be the same as [29] for fair comparison. The energy consumption measures are from CC2420 technical specification [30], and MAC parameters are from the BoX-MAC specification [3]. Each point in the graphs is a result of 1,000 simulation runs with different topologies and different seeds for generating random numbers.

4.2 Results

4.2.1 Impact of network density

In the first experiment, the number of nodes in the network is varied from 100 to 700. Since the simulation area is fixed, the number of nodes can be directly translated into network density. The traffic generation rate is three packets per second. For ORIA, the fixed packet hold time at each node is 5 s. The metrics measured are network lifetime and ratio of late packets. Network lifetime is defined as the time until the first node runs out of energy [32], and ratio of late packets is the percentage of packets that are arrived at the sink later than the delay requirement.

Figure 4 shows the result. First, the TREE protocol shows the worst lifetime because energy consumption is not balanced among nodes: packets always follow the same paths. TREE-D extends network lifetime by dynamically changing paths based on residual energy. The difference between TREE and TREE-D becomes larger as the node density increases, because there are more nodes to

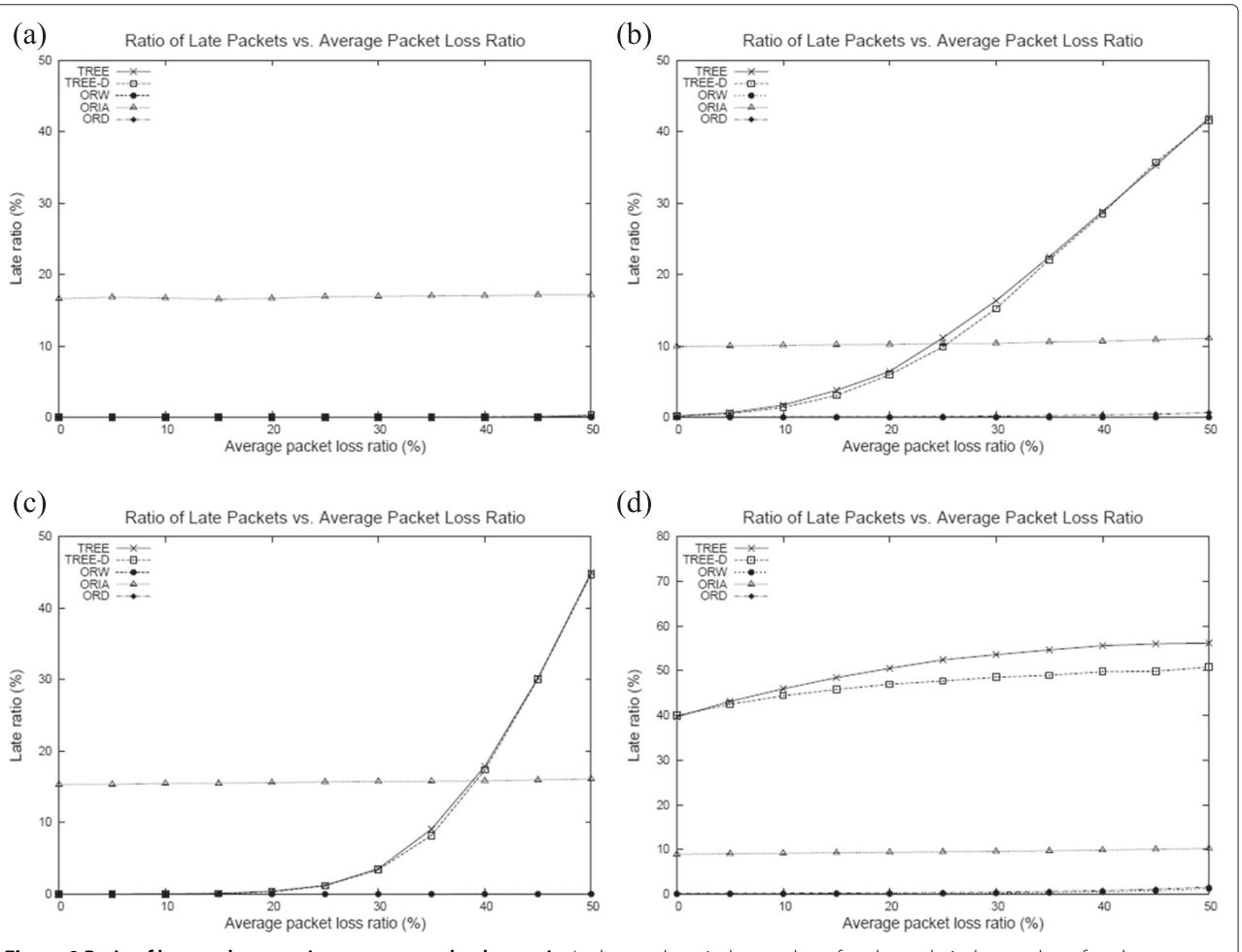

Figure 9 Ratio of late packets varying average packet loss ratio. In the graphs, n is the number of nodes and s is the number of packets generated per 30 seconds. **(a)** $n = 200$, $s = 20$. **(b)** $n = 200$, $s = 100$. **(c)** $n = 600$, $s = 60$. **(d)** $n = 600$, $s = 300$.

share the burden of forwarding packets. ORW achieves longer network lifetime by opportunistic forwarding. ORIA extends the lifetime further using in-network aggregation. However, it is shown in Figure 4b that 7% to 14% of packets are received at the sink late. This is caused by the fixed packet hold time of ORIA. ORD solves this problem by controlling packet hold time based on application delay requirements and thus achieves low late packet ratio. Also, ORD produces longer network lifetime compared to ORIA. This benefit is achieved from dynamically selecting forwarders (and thus packet hold times) based on energy status of the nodes.

4.2.2 Impact of traffic load

Next, the impact of traffic load on network lifetime and late packet ratio is studied by varying packet generation rate. The number of nodes is fixed at 200, and packet generation rate is varied from 5 to 100 packets per 30 seconds. Again, the fixed packet hold time is 5 s. The result is shown in Figure 5.

ORD outperforms TREE, TREE-D, ORW, and ORIA in terms of network lifetime. All protocols show similar pattern in which network lifetime decreases as the traffic load is increased, which is an intuitive result. When packet generation rate is low, network lifetime of ORIA is similar to ORW. This is because ORIA does not gain much from aggregation. ORD achieves 20% longer lifetime compared to ORIA with low traffic load. When the traffic load is high, network lifetime of ORD and ORIA is comparable, but ORIA produces 10% of late packets while achieving this lifetime. The late packet ratio of ORD is near zero, as shown in Figure 5b. To reduce the late packet ratio of ORIA, the packet hold time should be reduced, which in turn reduces network lifetime. The results with different packet hold times are presented later.

4.2.3 Impact of wake-up interval

In this experiment, wake-up interval is varied to study its impact on system performance. The results are obtained

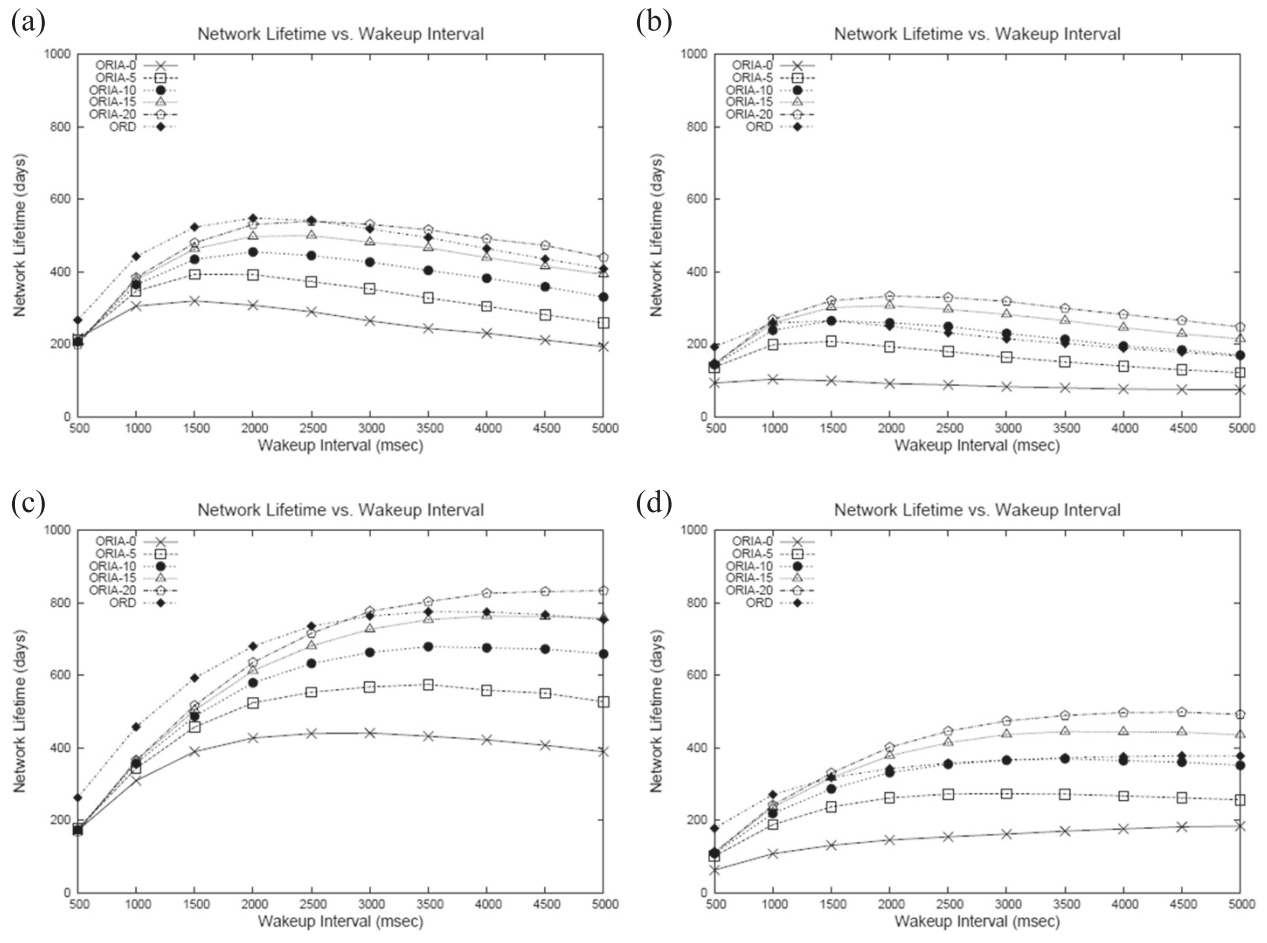

Figure 10 Network lifetime varying wake-up interval for comparing ORD and ORIA. In the graphs, n is the number of nodes and s is the number of packets generated per 30 seconds. **(a)** $n = 200$, $s = 20$. **(b)** $n = 200$, $s = 100$. **(c)** $n = 600$, $s = 60$. **(d)** $n = 600$, $s = 300$.

for four cases with different node densities and traffic loads and are shown in Figures 6 and 7.

In general, ORD achieves longer network lifetime than other protocols. Compared to ORIA, ORD achieves approximately 50% longer network lifetime. When the wake-up interval is short so is the network lifetime. As the wake-up interval increases, the network lifetime increases because nodes spend more time in sleep mode. At some point, however, the network lifetime decreases again. This is because if the wake-up interval is long, the wait time of senders becomes longer which brings down the network lifetime. Looking at Figure 7, it can be observed that late packet ratio increases rapidly for TREE and TREE-D protocols when the wake-up interval is high. For TREE and TREE-D, increase in wake-up interval directly means increased packet delay, since there is no opportunistic forwarding. For opportunistic protocols, late packet ratio does not increase much when the wake-up interval is increased. This benefit directly comes from opportunistic behavior. If there are n forwarders, c times increase in wake-up interval only translates into c/n times increase in the expected wait time.

4.2.4 *Impact of packet loss*

In this experiment, we have assigned a packet loss ratio for each link. Packet loss ratio for links is assigned according to normal distribution, centered at a given average. We vary this average in order to see how performance or protocols is affected by packet loss. The results are shown in Figures 8 and 9.

In each scenario, ORD achieves the highest network lifetime among all protocols. When the number of nodes is 200, network lifetime decreases as the packet loss ratio increases. This is because the sender has to send its packet longer if the packet is not received at the receiver. However, we can observe that when the network is very dense, the lifetime of opportunistic protocols can actually increase. When the network density is high, the chance that multiple receivers receive the packet becomes higher. When multiple receivers send back ACKs, they can collide

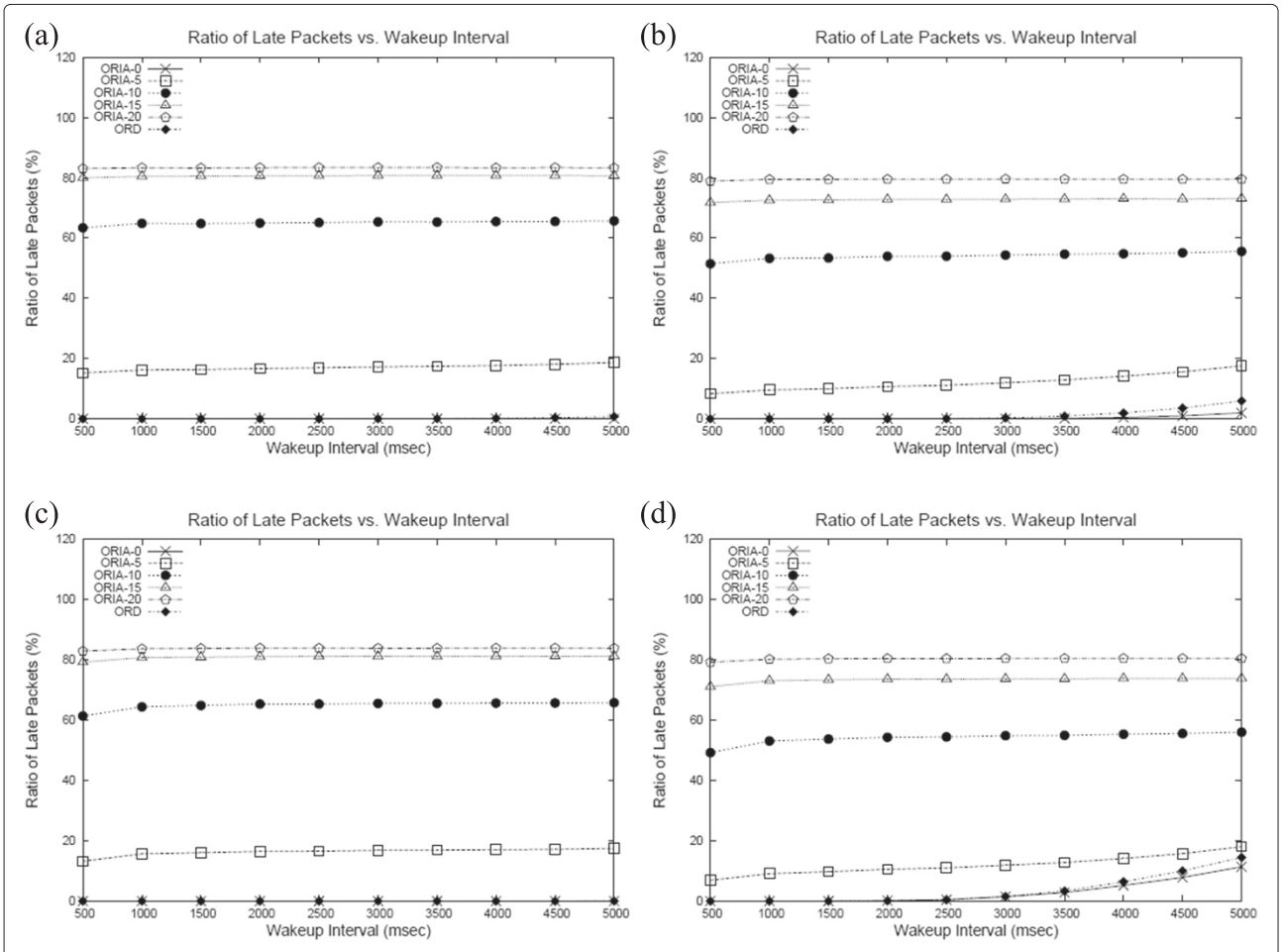

Figure 11 Ratio of late packets varying wake-up interval for comparing ORD and ORIA. In the graphs, n is the number of nodes and s is the number of packets generated per 30 seconds. **(a)** $n = 200$, $s = 20$. **(b)** $n = 200$, $s = 100$. **(c)** $n = 600$, $s = 60$. **(d)** $n = 600$, $s = 300$.

at the sender, and thus, the sender continues to send packets which results in unnecessary energy consumption. In this situation, packet loss can actually help towards reducing ACK collisions. If the benefit is greater than the negative effect, the network lifetime becomes longer. In terms of late ratio, it is not affected by packet loss when the traffic load is very low (s = 20). When the traffic load becomes higher, packet loss immediately affects the tree-based protocols, because they only maintain one path for each node. The opportunistic protocols are not significantly affected by loss rate, which is one of the major strengths of opportunistic protocols.

4.2.5 Comparing ORD and ORIA

Now, we directly compare ORD and ORIA protocols. For ORIA, the packet hold time was fixed at 5 s for the previous simulations. It is expected that network lifetime and late packet ratio significantly depend on the packet hold time for ORIA. In this simulation, the results are obtained for ORIA with packet hold time of 0, 5, 10, 15, and 20 s, as well as the ORD protocol. Note that ORIA with zero packet hold time is similar to ORW, but slightly different in that even if packet hold time is zero, there is still time gap between a node receiving a packet and actually forwarding the packet to another node. In this duration of time, the ORIA protocol applies short wake-up time. ORW does not have this mechanism.

The results are shown in Figures 10 and 11. For low traffic (Figure 10a,c), the network lifetime of ORD is comparable to ORIA with 15 or 20 s of hold time. For high traffic (Figure 10b,d), ORD is comparable to ORIA with 10 s of hold time. It can be observed that by increasing packet hold time of ORIA, one can achieve longer network lifetime. However, the late packet ratio depends significantly on packet hold time; 10% to 20% of packets are late with 5 s of hold time, and longer hold times are unacceptable when delay requirement is 30 s. One observation from Figure 11 is that when the traffic is very high, late ratio of ORW and ORD starts to increase. This is caused by high contention among nodes. Even

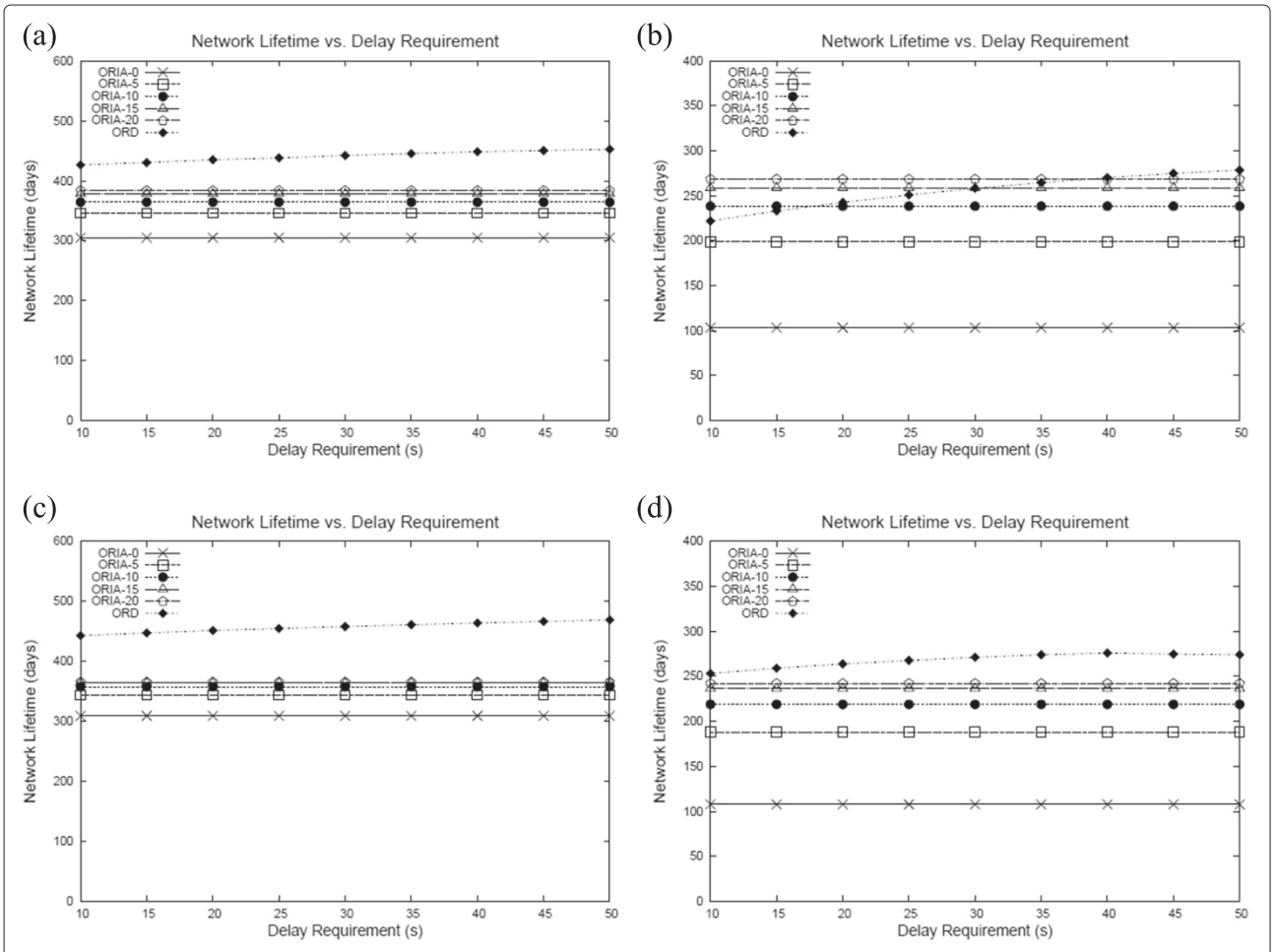

Figure 12 Network lifetime varying delay requirement for comparing ORD and ORIA. In the graphs, n is the number of nodes and s is the number of packets generated per 30 seconds. **(a)** n = 200, s = 20. **(b)** n = 200, s = 100. **(c)** n = 600, s = 60. **(d)** n = 600, s = 300.

ORW, which does not hold packets for in-network aggregation, fails to deliver packets on time for approximately 10% of packets. The late ratio of ORD is slightly higher because the packets may spend some time in the downstream nodes before getting caught in high contention near the sink. In moderate environments, ORD and ORIA with zero hold time achieves near-zero late packet ratio, and network lifetime of ORD is significantly longer than ORIA.

Finally, the delay requirement is varied to see if ORD meets delay requirements well. Delay requirement is varied from 10 to 30 s, and ORIA with different hold times is compared with ORD. The results are shown in Figures 12 and 13. The network lifetime of ORIA protocols is not affected by delay requirements, whereas network lifetime of ORD increases as the delay requirement is increased. This shows that ORD can benefit from setting delay requirement large, if this is tolerable with the applications. ORD achieves this benefit by holding packets longer until time permits, so that more packets can be aggregated. As shown in Figure 13, ORD successfully meets the delay requirement even when the delay requirement is 10 s. Whereas 5 s of hold time in ORIA leads to 80% of late packets with 10 s delay requirement.

In summary, ORD successfully benefits from tolerance of applications for packet delays, by extending network lifetime through in-network aggregation. However, limitations and rooms for improvement still exist, and it has got to do with wake-ups. First, ORW, ORIA, and ORD all assume that nodes run duty cycles using the same wake-up intervals. However, wake-up intervals can be controlled to manipulate probabilities of packet reception among forwarders. Since nodes placed near the sink tend to drain energy quicker, wake-up intervals can be controlled so that packets travel faster when they are far away from the sink, and take time near the sink so that energy consumption is balanced, and thus, network

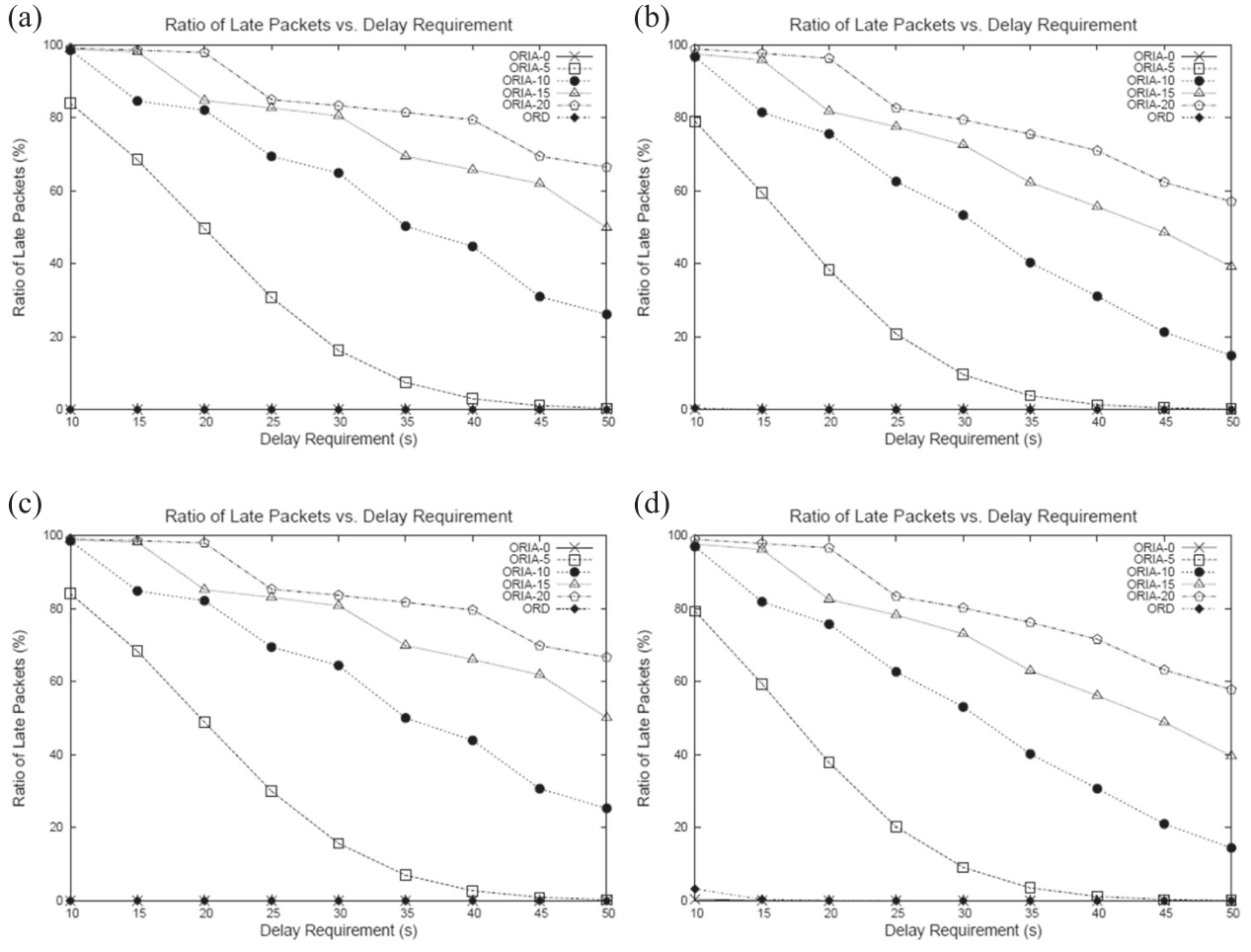

Figure 13 Ratio of late packets varying delay requirement for comparing ORD and ORIA. In the graphs, *n* is the number of nodes and *s* is the number of packets generated per 30 seconds. **(a)** $n = 200$, $s = 20$. **(b)** $n = 200$, $s = 100$. **(c)** $n = 600$, $s = 60$. **(d)** $n = 600$, $s = 300$.

lifetime is extended. Second, the short wake-up intervals used to increase the chance of reception for nodes holding packets are not optimized. Since it is a source of energy consumption, the short wake-up intervals should be carefully chosen based on the probability of receiving another packet from downstream nodes. These issues will be studied as a future work.

5 Conclusions

In this paper, we propose ORD, an opportunistic routing protocol for data collection in wireless sensor networks, where applications have certain requirements on packet delays. ORD shares basic features with ORW where forwarder sets are computed based on EDC, and nodes take advantage of multiple candidate forwarders and their random wake-up times to quickly transfer their packets. ORD follows ORIA's approach where nodes hold on to their packets before forwarding, and meanwhile shorten their sleep periods to aggregate packets from downstream nodes. However, unlike ORIA, ORD uses feedback information on residual energy levels and maximum hop distance in order to control packet hold times, so that the network lifetime is maximized while packet delay requirements are met. Simulation results show that ORD outperforms ORIA in both network lifetime and ratio of late packets. In summary, ORD provides network operators with a tuning knob which controls target packet delay and optimizes parameters accordingly so that the operators do not need to specify protocol parameters such as packet hold time for each node.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This work is supported by the Hallym University Research Fund 2014 (HRF-201401-013).

Author details
[1]Department of Computer Engineering, Hallym University, 1 Hallymdaehak-gil, Chuncheon 200-702, South Korea. [2]Department of Information and Telecommunications Engineering, Suwon University, San 2-2 Wau-ri, Bongdam-eup, Hwaseong 445-743, South Korea.

Received: 21 September 2014 Accepted: 3 December 2014
Published: 9 December 2014

References

1. J Polastre, J Hill, D Culler, in *ACM Sensys*. Versatile low power media access for wireless sensor networks, (2004), pp. 95–107
2. M Buettner, G Yee, E Anderson, R Han, in *ACM Sensys*. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks, (2006), pp. 307–320
3. D Moss, P Levis, *Exploiting physical and link layer boundaries in low-power networking*. (Stanford University Technical Report SING-08-00. (2008)
4. P Levis, S Madden, J Polastre, R Szewczyk, K Whitehouse, A Woo, D Gay, J Hill, M Welsh, E Brewer, D Culler, in *Ambient Intelligence*. Tinyos: An operating system for sensor networks, (2005), pp. 115–148
5. O Gnawali, R Fonseca, K Jamieson, D Moss, P Levis, in *ACM Sensys*. Collection tree protocol, (2009), pp. 1–14
6. O Landsiedel, E Ghadmi, S Duquenoy, M Johansson, in *IPSN*. Low power, low delay: opportunistic routing meets duty cycling, (2012), pp. 185–196
7. A Bachir, M Dohler, T Watteyne, K Leung, MAC essentials for wireless sensor networks. IEEE Commun. Surv. Tutor. **12**(2), 222–248 (2010)
8. K Han, J Luo, AV Vasilakos, Algorithm design for data communications in duty-cycled wireless sensor networks: a survey. IEEE Commun. Mag. **51**(7), 107–113 (2013)
9. IF Akyildiz, T Melodia, KR Chowdhury, A survey on wireless multimedia sensor networks. Elsevier Comput. Netw. **51**, 921–960 (2007)
10. J So, J Kim, I Gupta, in *IEEE MASS (poster)*. Cushion: autonomically adaptive data fusion in wireless sensor networks, (2005)
11. SC Ergen, P Varaiya, Energy efficient routing with delay guarantee for sensor networks. Springer Wireless Netw. **13**(5), 679–690 (2007)
12. W Ye, J Heidemann, D Estrin, in *IEEE INFOCOM*. An energy efficient MAC protocol for wireless sensor networks, (2002), pp. 1567–1576
13. T van Dam, K Langendoen, in *ACM Sensys*. An adaptive energy-efficient MAC protocol for wireless sensor networks, (2003), pp. 171–180
14. P Lin, C Qiao, X Wang, in *WCNC*. Medium access control with a dynamic duty cycle for sensor networks, (2004), pp. 1534–1539
15. D De Couto, D Aguayo, J Bicket, R Morris, A high-throughput path metric for multi-hop wireless routing. Springer Wireless Netw. **11**(4), 419–434 (2005)
16. S Biswas, R Morris, in *ACM SIGCOMM*. ExOR: opportunistic multihop routing for wireless networks, (2005), pp. 133–144
17. M Zorzi, R Rao, Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Multihop performance. IEEE Trans. Mobile Comput. **2**(4), 337–348 (2003)
18. Y Gu, T He, in *ACM SenSys*. Data forwarding in extremely low duty-cycle sensor networks with unreliable links, (2007), pp. 321–334
19. E Fasolo, M Rossi, J Widmer, M Zorzi, In-network aggregation techniques for wireless sensor networks: a survey. IEEE Wireless Commun. **14**(2), 70–87 (2007)
20. B Krishnamachari, D Estrin, S Wicker, in *IEEE INFOCOM*. Modelling data-centric routing in wireless sensor networks, (2002)
21. T Aonishi, T Matsuda, S Mikami, H Kawaguchi, C Ohta, M Yoshimoto, in *IEEE International Conference on Parallel Processing Workshops*. Impact of aggregation efficiency on GIT routing for wireless sensor networks, (2006), pp. 151–158
22. C Liu, G Gao, in *IEEE INFOCOM*. Distributed monitoring and aggregation in wireless sensor networks, (2010), pp. 1–9
23. L Villas, A Boukerche, H Ramos, H de Oliveira, R de Araujo, A Loureiro, Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. IEEE Trans. Comput. **62**(4), 676–689 (2013)
24. Y Yao, Q Cao, AV Vasilakos, Edal: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks. IEEE/ACM Trans. Netw. **99**, 1-1 (2014)
25. N Chilamkurti, S Zeadally, A Vasilakos, V Sharma, Cross-layer support for energy efficient routing in wireless sensor networks. J. Sensors. **2009**(article id 134165) (2009)
26. L Xiang, J Luo, AV Vasilakos, in *IEEE SECON*. Compressed data aggregation for energy efficient wireless sensor networks, (2011), pp. 46–54
27. G Wei, Y Ling, B Guo, B Xiao, AV Vasilakos, Prediction-based data aggregation in wireless sensor networks combining grey model and kalman filter. Comput. Commun. **34**(6), 793–802 (2011)
28. X-Y Liu, Y Zhu, L Kong, C Liu, Y Gu, AV Vasilakos, M-Y Wu, Cdc: Compressive data collection for wireless sensor networks. IEEE Trans. Parallel Distributed Syst. **99**(1) (2014)
29. J So, H Byun, Opportunistic routing with in-network aggregation for asynchronous duty-cycled wireless sensor networks. Springer Wireless Netw. **20**(5), 833–846 (2014)
30. AS Chipeon, SmartRF CC2420 Preliminary datasheet (rev 1.2) (2004)
31. R Jurdak, A Ruzzelli, G O'Hare, in *IEEE SECON*. Adaptive radio modes in sensor networks: how deep to sleep? (2008), pp. 386–394
32. I Dietrich, F Dressler, On the lifetime of wireless sensor networks. ACM Trans. Sensor Netw. (TOSN). **5**(1) (2009). article no. 5
33. Mo Li, et al., A Survey on Topology Control in Wireless Sensor Networks: Taxonomy, Comparative Study, and Open Issues. Proceedings of the IEEE. **101**(12), 2538–2557 (2013)

34. M Youssef, et al., Routing Metrics of Cognitive Radio Networks: A Survey. IEEE Communications Surveys and Tutorials. **16**(1), 92–109 (2014)
35. Zeng Yuanyuan, et al., Directional routing and scheduling for green vehicular delay tolerant networks. Wireless Networks. **19**(2), 161–173 (2013)

doi:10.1186/1687-1499-2014-217
Cite this article as: So and Byun: **Opportunistic routing with in-network aggregation for duty-cycled WSNs with delay requirements.** *EURASIP Journal on Wireless Communications and Networking* 2014 **2014**:217.