This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

Load-Balanced Opportunistic Routing for Duty-Cycled Wireless Sensor Networks

Jungmin So and Heejung Byun, *Member, IEEE*

Abstract—In duty-cycled wireless sensor networks running asynchronous MAC protocols, the time when a sender waits for its receiver to wake up and receive the packet is the major source of energy consumption. Opportunistic routing can reduce the sender wait time by allowing multiple candidate receivers, but by doing that it suffers from redundant packet forwarding due to multiple receivers waking up at the same time. Thus, number of forwarders should be controlled in a way that overall forwarding cost is minimized considering both sender wait time and cost of redundant packet forwarding. Also, in order to prolong network lifetime, candidate forwarders should be selected so that load is balanced among nodes. We propose ORR, an opportunistic routing protocol that addresses the two issues. First, optimal number of forwarders is calculated based on forwarding cost estimation, which is derived from duty cycle and network topology. Second, the metric used for selecting forwarders considers residual energy so that more traffic is guided through nodes with larger remaining energy. The resulting routing protocol is proven to avoid loops and shown to achieve longer network lifetime compared to other protocols regardless of duty cycle and network topology.

Index Terms—wireless sensor networks, opportunistic routing, load balancing.

1 Introduction

Energy efficiency is one of the most important factors to consider when designing protocols for wireless sensor networks (WSNs). Energy is important not only because it can be costly to exchange batteries or add new sensor nodes in the target area, but because sometimes it is very difficult or even impossible to maintain control over the nodes, such as networks deployed deep inside ocean, near an active volcano, or a battlefield. Medium access control (MAC) and routing protocols designed for WSNs try to achieve long network lifetime while supporting application requirements.

Duty-cycling is a method commonly used in WSNs to trade packet delay with network lifetime. When duty-cycling is used, nodes switch between active and sleep modes according to a predefined wake-up interval. While in sleep mode nodes cannot send or receive packets, but spend much less energy compared to the active mode [1].

When a pair of nodes want to communicate, they both have to be in the active mode. To coordinate node states, a MAC protocol can follow one of the two approaches: synchronous and asynchronous. In a synchronous MAC protocol, wake-up time of nodes is synchronized so that multiple nodes wake up at the same time. Since active period is synchronized, a sender can assume that the receiver is also active and is ready to receive a packet. Problem with synchronized protocols is that message overhead is needed in order to maintain time synchronization among nodes. Also, a mechanism such as random backoff is needed to resolve packet collisions, since multiple nodes may start transmission at the beginning of an active period. On the other hand, nodes wake up at different times in an asynchronous protocol. So either sender or receiver needs to wait in the active mode until the counterpart wakes up, and this wait time is the major source of energy consumption in asynchronous protocols. Compared to synchronized protocols, asynchronous protocols are simple and easy to implement.

When using an asynchronous MAC protocol, the wait time of sender or receiver significantly affects network lifetime. For example, when running BoX-MAC [2], the sender has to stay in active mode transmitting packets until the receiver wakes up and receives the packet. Depending on the node schedules, the sender may have to wait for a long time in the active mode. Opportunistic routing approach can be effective in reducing sender wait time by allowing multiple candidate forwarders instead of a single forwarder. When the sender transmits a packet, any node in the sender's forwarder set can receive and forward the packet. This approach particularly fits well with asynchronous MAC protocols, since the sender can end its transmission when the first node receives the packet and sends back an acknowledgment (ACK). ORW (Opportunistic Routing in Wireless sensor networks) [3] is a protocol that uses this approach to achieve low energy consumption.

Although allowing multiple forwarders can reduce sender wait time, it can also generate duplicate copies of packets causing unnecessary energy consumption. If a packet is received by multiple forwarders, each of them will forward the same packet towards the sink. Moreover, since their ACKs collide at the sender, the sender continues to transmit its packet extending the sender wait time. If too many duplicate packets are generated in the network, energy consumed for delivering unnecessary copies of packets can exceed the amount of energy saved by reducing sender wait time. In that case, the opportunistic routing may

- *J. So is with the Dept. of Computer Engineering, Hallym University, Chuncheon, Korea. E-mail: jso1@hallym.ac.kr*
- *H. Byun (corresponding author) is with the Dept. of Information and Telecommunications Engineering, Suwon University, Hwaseong, Korea. E-mail: heejungbyun@suwon.ac.kr*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

perform even worse than conventional tree-based routing.

The amount of packet duplication depends on duty cycle and number of forwarders. When duty cycle is very low, the chance of multiple forwarders receiving the same packet is small. However, as duty cycle becomes higher, packet duplication starts to pose a significant impact on the protocol performance. Similarly, the chance of packet duplication is higher when nodes have more candidate forwarders. Thus, either duty cycle or number of forwarders should be controlled so that the energy consumption is minimized. In this paper, we focus on controlling number of forwarders, because duty cycle may need to be chosen based on application requirements such as packet latency.

In this paper we propose ORR, an opportunistic routing protocol for asynchronous duty-cycled WSNs. ORR uses the same approach as ORW, but improves upon ORW in the following ways. First, ORR calculates optimal number of forwarders based on forwarding cost estimation. The optimal number of forwarders can differ according to network environment, so it needs to be calculated on-line during operation. Forwarding cost estimation considers expected sender wait time and expected amount of redundant packets generated. Second, residual energy is considered when nodes select their forwarder sets. In ORW, forwarders are selected based on expected wait time, so nodes with large number of neighbors have higher chance of becoming candidate forwarders. Thus, traffic load is often concentrated at small number of nodes, draining their energy faster than other nodes. In ORR, nodes with larger residual energy become forwarders more often. Algorithm for calculating optimal number of forwarders is presented using mathematical analysis. Performance of ORR is evaluated through extensive simulations, comparing various widely used protocols and exploring impact of various parameters.

Rest of the paper is organized as follows. Section 2 discusses existing schemes that are relevant to our work. Section 3 describes relevant protocols such as BoX-MAC and ORW, and motivates our work by discussing potential problems of ORW. Section 4 presents the proposed protocol, which addresses the discussed problems. Section 5 evaluates performance of the proposed protocol using extensive simulation. Section 6 discusses issues and limitations of the proposed protocol. Finally, Section 7 concludes the paper.

2 Related Work

For more than a decade, a large number of protocols have been proposed for wireless sensor networks. Some protocols are adopted from other types of wireless networks such as wireless LANs, but constraints on energy makes protocol design for WSNs unique [4]. Medium access control and routing protocols for WSNs have a common goal of minimizing energy consumption and thus achieving a long network lifetime.

Most MAC protocols for WSNs are built on top of duty-cycling, and they can be divided into synchronized and asynchronous MAC. SMAC [5] is the most widely known synchronized MAC protocol. Nodes are time-synchronized so that they wake up at the same time. A node can send messages while in active mode, because other nodes are expected to be in active mode as well. Nodes exchange SYNC messages in order to maintain time synchronization, and RTS/CTS mechanism is used to avoid collisions. Many other protocols follow this approach and try to achieve their goals using common active periods [6], [7], [8], [9].

On the other hand, asynchronous MAC protocols allow nodes to wake up at different times, and thus there is no need for time synchronization. Instead, in order for two nodes to communicate, either the sender or the receiver has to wait for its counterpart to wake up. For example, in B-MAC [10], sender transmits a preamble for the entire wake-up interval, and nodes receiving the preamble stay in the active mode. After waking up the neighbors, the sender sends its packet. X-MAC [11] improves on B-MAC by sending short preambles repeatedly instead of a long preamble. The short preamble contains destination information, so a node can identify if the transmission is destined to itself. If so, the node sends an ACK back to the sender. The sender then transmits its packet to the receiver. X-MAC's improvement from B-MAC is that the sender only needs to transmit preamble until the receiver wakes up, and that only the destination node has to stay up when it receives the preamble. BoX-MAC [2], which is adopted as the default MAC protocol in TinyOS [12], sends the packet repeatedly instead of the preamble. When the destination node receives the packet and sends back an ACK, the sender stops transmission and the communication ends there. In addition to these main protocols, other protocols were proposed that follow the asynchronous wake-up approach [13], [14], [15].

Although X-MAC and BoX-MAC prevent senders from staying in active mode for the whole wake-up interval, a sender still needs to stay in active mode until its receiver wakes up. When there is a single forwarder for each node as in collection trees [16], the expected wait time is half of a wake-up interval, which can be considered long. Opportunistic routing is a good strategy to further reduce the sender wait time. Instead of having a single receiver, opportunistic routing allows multiple receivers, in which any of the node can receive and forward the sender's packet. In ExOR [17], each node has a prioritized list of candidate forwarders, which is included in the packet header before the packet is transmitted. When a node receives a packet, it forwards the packet but after a delay calculated from the priority. If the node overhears the same packet being forwarded by another forwarder, it drops the packet. ExOR assumes that all nodes are in active mode and they can overhear each other's packets, which is not the case for duty-cycled WSNs. GeRaF [18] is similar to ExOR, but the forwarder list is selected and prioritized based on geographical location of the forwarders. EQGOR [19] also uses geographical location but prioritizes forwarder sets based on QoS requirements such as reliability and end-to-end delay. DSF [20] is an opportunistic routing protocol designed to work with synchronized MACs. Forwarding nodes are selected based on sleep schedules of neighbors, as well as delay, reliability and energy consumption. Control messages should be exchanged between nodes in order to find out the sleep schedules of neighboring nodes, which is an overhead for this protocol.

ORW [3] is an opportunistic routing protocol that is designed to work on top of an asynchronous MAC protocol. Instead of a tree, ORW builds a DODAG (Destination

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

Oriented Directed Acyclic Graph) in the network using a metric called EDC (Expected Duty Cycled Wakeups). Forwarders are selected so that when forwarded to one of them, the packet advances towards the sink meaning that the expected amount of delay until the packet reaches the sink is reduced. Since ORW is the protocol that inspired the proposed scheme, it is explained in detail in the next section. One problem that stands out i n ORW is the *multiple receiver problem*, which degrades performance by generating redundant packets. DOF [21] introduces a scheme that can detect and remove duplicates so that unnecessary forwarding is prevented and energy is conserved. Instead of trying to detect and remove duplicates, our proposed protocol tries to optimize energy consumption by controlling number of forwarders. ORIA [22] improves the network lifetime of ORW by promoting in-network aggregation. If a node receives a packet to forward, it holds the packet for a certain amount of time, and in the meantime increases its duty-cycle in order to increase probability of receiving other packets. When multiple packets are gathered, the node performs data fusion and combines multiple packets into one, thereby reducing number of transmissions. Although ORIA achieves higher network lifetime compared to ORW, packet delay is significantly increased because of the hold time.

There are various protocols that try to achieve long network lifetime using load balancing. In tree-based routing protocols, load balancing is achieved by selecting paths based on residual energy of nodes in the path [23]. In cluster-based collection protocols, cluster heads dynamically change so that nodes with larger remaining energy become cluster heads [24], [25]. In systems with duty-cycling, duty cycle can be chosen based on the residual energy so that nodes with higher residual energy wake up more often than nodes with lower residual energy [26]. For opportunistic routing protocols, load balancing can be achieved by selecting forwarders based on residual energy. In the proposed protocol, forwarder selection metric is designed so that nodes with higher residual energy become forwarders more often.

3 Preliminaries

3.1 BoX-MAC and ORW

BoX-MAC [2] is the default MAC protocol in TinyOS [12]. It is an asynchronous MAC protocol in which nodes are not time-synchronized. So when a sender transmits its packet, its intended receiver may be in sleep mode. Thus in BoX-MAC, a node transmits its packet repeatedly until its receiver wakes up and receives the packet. When a node wakes up, it checks the channel to see if there is any packet being transmitted. If the channel is idle, the node goes back to sleep. If there is a packet on the channel, the node performs layer 2 receive check to see if the node should receive the full packet. Depending on this receive check, the duration of active mode is determined for the receiver. If the node receives the packet, it sends back an ACK to the sender, and the sender stops transmitting on receving the ACK. The behavior of BoX-MAC is described in Fig. 1.

The major source of energy consumption in BoX-MAC is *sender wait time*, the duration of time a sender transmits its packet until the receiver sends back an ACK. With unicast

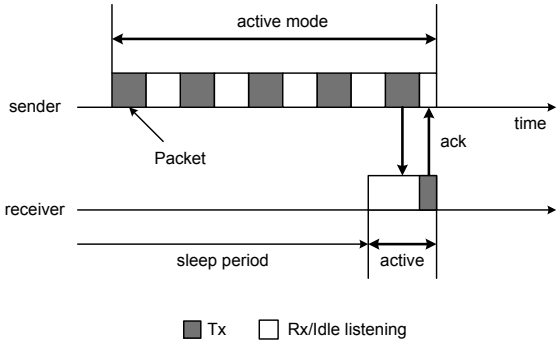


Fig. 1. Behavior of BoX-MAC. The sender repeatedly transmits packets, with time gaps between packets to receive ACKs. When the receiver receives the packet, it sends back an ACK. The sender stops transmission and goes back to sleep when it receives an ACK for the packet.

routing, a node has to wait for a half of a wake-up interval in average. Opportunistic routing can help reduce sender wait time by allowing multiple candidate receivers. ORW [3] is a routing protocol that follows this approach. In ORW, each node maintains a *forwarder set*. Any node in the forwarder set can receive and forward the sender's packet. Suppose nodes B, C, and D are in the forwarder set of node A. When node A has a packet to send, it switches to active mode and starts transmitting the packet. Among B, C, and D, node B wakes up first. So it receives A's packet and sends an ACK back to A. On receiving the ACK, node A stops transmission. Node C and D cannot hear A's packet since it wakes up after B. This behavior is illustrated in Fig. 2.

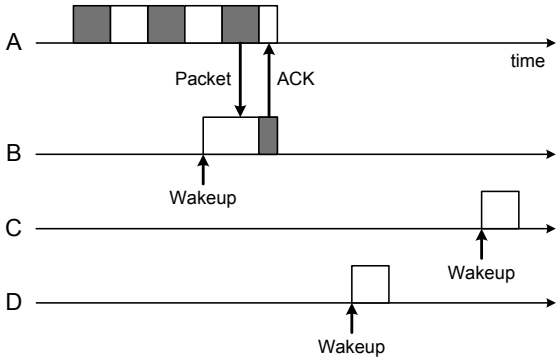


Fig. 2. Behavior of ORW. Nodes B, C, and D are neighbors of node A, and they are in the forwarder set of node A. After node A starts transmitting its packet, node B wakes up first, receives the packet and sends back an ACK. Node A finishes transmission when it receives an ACK from node B.

Nodes in the forwarder set are selected from neighbors. Not all neighbors should be included in the forwarder set, because some neighbors may be "more distant" from the sink. Here, distance may not only mean physical distance, but it may be defined in terms of time or energy. Under the given definition of distance, a forwarder should provide "forwarding progress" to the packet, meaning the packet is actually advancing towards the sink.

In ORW, forwarder sets are selected using the EDC metric. EDC is a measure of expected duration of time (in number of wake-up periods) until a packet reaches its destination. If EDC is 1, it means that the packet is expected

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

TABLE 1
Notations used in the paper.

Notation	Description
F_i	forwarder set of node i.
n_i	number of forwarders in F_i.
n_{max}	maximum number of forwarders.
ν_{ij}	delivery ratio of the link between node i and j.
q	probability of multiple receivers receiving the sender's packet.
T_a	active period.
T	wake-up interval.
S	number of time slots in a wake-up interval.
p_c	probability of ACK collision.
$p_{n,S}(k)$	probability that one of n forwarders wake up at the kth slot (out of S slots) and all other nodes wake up at later slots.
ζ_i	quantized energy level of node i.
E_i, E_0	residual energy of node i and initial energy.
$t_{n,S}$	expected sender wait time with n forwarders and S slots in a wake-up interval.
$\varphi_{n,S}(l,m)$	probability that l nodes wake up before the m-th slot and a single node wakes up in the m-th slot, with n forwarders and S slots.
$\vartheta_{l,m}$	probability that transmission is not successful within m slots, when l forwarders wake up during that time.
$R_{n,S}$	expected number of received packets with n forwarders and S slots in a wake-up interval.
Q_i	expected number of packet transmissions from node i.

to reach the sink node in the duration of a single wake-up period. EDC is calculated using Eq. (1). EDC_i indicates the EDC of node i. Notations used in this equation and other equations in the paper are summarized in Table 1.

$$EDC_i = \frac{1}{\sum\limits_{j \in F_i} \nu_{ij}} + \frac{\sum\limits_{j \in F_i} \nu_{ij} EDC_j}{\sum\limits_{j \in F_i} \nu_{ij}} + \omega_0. \quad (1)$$

Eq. (1) consists of three terms. The first term is called single-hop EDC, which is determined by the node's own forwarder set. The second term accounts for the path where the packet travels after it is forwarded to one of the candidate forwarders. The third term ω_0 is a constant which is added to account for the cost of forwarding.

Each node computes its forwarder set based on the neighbor information. Starting from an empty set, neighbor nodes are inserted into the forwarder set in the ascending order of their EDCs, and EDC of the node is recalculated every time a new neighbor is inserted. The process stops when the EDC of the next neighbor is larger than the current EDC of the node. In the structure established by EDC, it is possible that a node with larger hop distance from the sink can become a forwarder of a node with shorter hop distance. Thus, it is very important that the forwarding structure does not create a loop. The forwarding structure is proven to have a loop-free characteristics, since the graph established by forwarder sets form a DODAG (Destination Oriented Directed Acyclic Graph). An upstream node in the DODAG structure always has a smaller EDC compared to a downstream node.

3.2 Link quality estimation problem

Here we discuss the use of estimated link quality in forwarder selection. ORW includes link quality term in EDC, as shown in Eq. (1). A node that has links with higher quality will have smaller EDC, because the expected sender wait time is shorter. In ORW, link quality is estimated using overhearing [3]. Packets contain a header field that denotes the average rate at which a node is forwarding data. Link quality is obtained by dividing the rate of packets overheard from a neighbor by the forwarding rate of the same neighbor noted in the header field.

The assumption of the scheme is that if the quality of link between node A and C is 100%, node A should overhear every packet node C transmits. However, this is not true, as can be seen from Fig. 2. C does not overhear A's packet because B wakes up and receives the packet before C wakes up. Probability that node C will overhear A's packet depends on A's number of forwarders, as shown in Fig. 3.

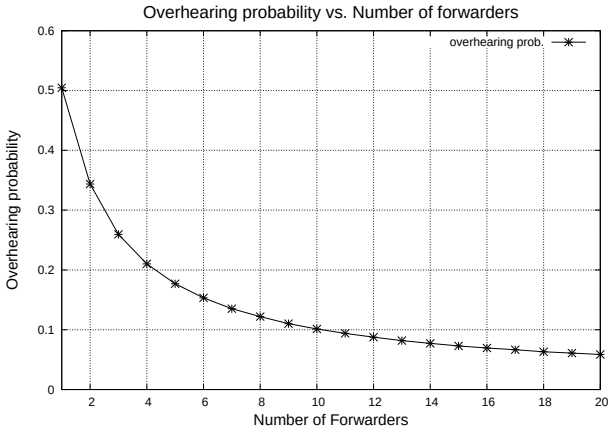


Fig. 3. Probability that node C overhears node A's packet. The X-axis is A's number of forwarders. 100% link quality is assumed.

If node A has 10 forwarders, node C will only overhear 10% of A's packets. So if node C estimates quality of link between A and C using overhearing, it will lead to a significantly inaccurate estimation. Inaccurate link quality estimation can result in poor choice of forwarders, and negatively affect the performance.

For this reason, we need a better estimation scheme that is accurate and cost-effective. Otherwise, it is better not to include link quality in the forwarder selection metric. If we remove the link quality term, the EDC equation can be simplified as:

$$EDC_i = \frac{1}{n_i + 1} + \frac{\sum\limits_{j \in F_i} EDC_j}{n_i}. \quad (2)$$

Note that 1 is added to denominator of the first term, because it is a more accurate model for the expected wait

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

time. If a node has a single forwarder, the expected wait time is 0.5 (in unit of wake-up interval). If the node has two candidate forwarders the expected wait time is 0.33, and so on. The constant ω_0 is also omitted for simplicity.

Due to inaccuracy of estimation scheme, we do not include the link quality term in the metric used for the proposed protocol. However, link quality may not be 100% in real life due to weak signal, fading, and interference. In Section 6, we discuss techniques for dealing with low-quality links. Note that if an accurate and cost-effective method can be developed, it is always possible to include the link quality term in the metric without difficulty.

3.3 Multiple receiver problem

3.3.1 Negative effects of multiple receivers

Sender wait time is expected to decrease with increasing number of forwarders, as also expressed in the first term of Eq. (2). However, if there are a large number of forwarders, it is possible that multiple receivers wake up at similar times to receive the same packet, as shown in Fig. 4.

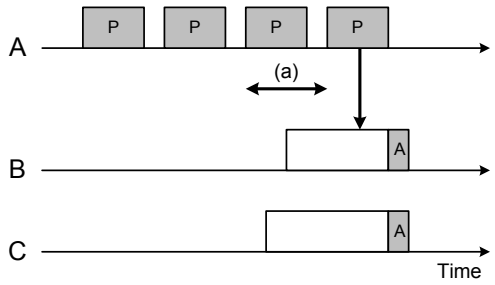


Fig. 4. A scenario illustrating multiple receiver problem. Node B and C receives A's packet at the same time and both send ACKs to node A. Nodes that wake up during period (a) will receive the same packet.

We call this the *multiple receiver problem*, and it can negatively affect energy consumption of the network in two ways. First, duplicate packets are generated in the network which will be forwarded independently up to the sink. Second, the sender fails to receive an ACK due to collision and continues to transmit packets, causing energy waste at the sender and generating even more duplicate packets.

When the number of forwarders increases, the expected sender wait time may decrease but the chance of multiple receiver problem increases. Duty cycle also has the same effect: high duty cycle leads to shorter sender wait time because nodes wake up more frequently, but also increases chance of multiple receiver problem. Since duty cycle may be decided based on application requirements such as packet delay and report period, we focus on the number of forwarders. Since EDC does not consider the negative effect, nodes tend to choose a large number of forwarders. But in order to minimize energy consumption, number of forwarders should be selected considering both the positive and negative effects.

ORW provides three light-weight techniques to mitigate the multiple receiver problem. They are probabilistic ACK, overhearing, and network-layer duplicate detection. Although these techniques can help removing some duplicate packets, they are not fundamental solutions to the problem. These techniques are further discussed In Section 6.

3.3.2 Probability of multiple receivers

The probability of multiple receivers q is discussed in ORW [27], where it is described as:

$$q = 1 - \left(1 - \frac{T_a}{T}\right)^{n-1}, \tag{3}$$

where n is the number of forwarders. But this equation is misleading, because it is the probability that a node wakes up at a certain time slot (of duration T_a) while all other nodes wake up at a different slot. The equation redundantly counts cases where multiple nodes wake up at unique time slots. The actual multiple receiver probability is smaller.

Suppose node A has n nodes in its forwarder set. Also, we denote T/T_a in Eq. 3 as S, which is the number of time slots in a wake-up period. Once node A starts transmitting its packet, n candidate forwarders will wake up in any of the S slots, starting from the slot where A begins transmission. On each slot, three cases are possible: (1) no forwarder wakes up, (2) a single forwarder wakes up, or (3) two or more forwarders wake up. A multiple receiver problem occurs on ith slot if no forwarder wakes up until $(i-1)$th slot and two or more forwarders wake up in ith slot.

Now we calculate the probability of multiple receivers, p_c. First, the probability that two nodes wake up at ith slot and all other nodes wake up after ith slot is:

$${}_nC_2 \times \left(\frac{1}{S}\right)^2 \times \left(1 - \frac{i}{S}\right)^{n-2}. \tag{4}$$

It is possible that more than two nodes wake up on the ith slot. So we need to add the probability of k nodes waking up at ith slot and other nodes waking up later than the ith slot, in order to obtain the probability that a collision occurs on the ith slot. That is:

$$\sum_{k=2}^{n} {}_nC_k \times \left(\frac{1}{S}\right)^k \times \left(1 - \frac{i}{S}\right)^{n-k}. \tag{5}$$

Thus, p_c can be written as:

$$p_c = \sum_{i=1}^{S} \sum_{k=2}^{n} {}_nC_k \left(\frac{1}{S}\right)^k \left(1 - \frac{i}{S}\right)^{n-k}. \tag{6}$$

In this analysis, a slot-based model is used where time is divided into slots and each node chooses a random wake-up slot. In reality, nodes can wake up at any time. Still, the slot-based model fits well when calculating probability of multiple receivers. In Fig. 4, although node B and C wake up at different times, they receive the same packet and their ACKs collide at node A. In the analysis, we consider node B and C as if they wake up at the same slot.

3.3.3 Comparison of analysis and simulation

The multiple receiver probability can also be obtained by simulation. In the simulation, each of n forwarders randomly chooses a slot from the range [1, S]. On each try, slots are checked starting from the first slot to see if any node has chosen the slot. If the slot is selected by none, the next slot is checked. If the slot is selected by a single node, this try is marked as "single receiver". If the slot has multiple nodes, then this try is counted as "multiple receivers". For

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

a particular number of forwarders, 10,000,000 tries were conducted, and the ratio of multiple receivers is calculated. In Fig. 5, probability of multiple receivers calculated by the original equation (Eq. (3)), the modified equation (Eq. (6)), and the simulation are plotted. In this experiment, the active period is 50ms and the wake-up interval is 1 second, so the number of slots in a wake-up period is 20. From the

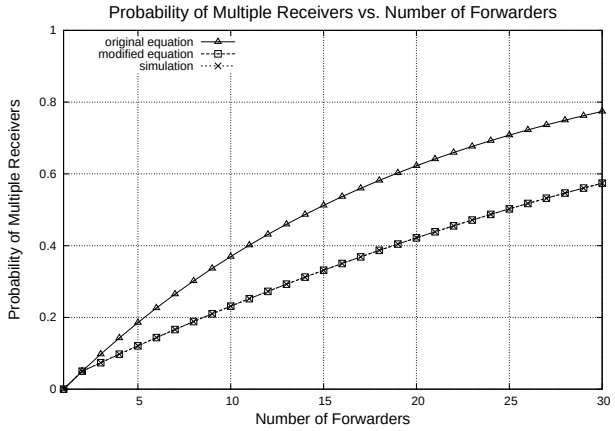


Fig. 5. Probability of multiple receivers varying number of forwarders.

figure, we can observe that multiple receiver probability is larger than 20% with 10 forwarders, and the probability is larger than 40% with 20 forwarders. The actual number of forwarders calculated by EDC will vary according to environmental factors such as node density, but as we can see in the evaluations, performance of ORW suffers significantly from this multiple receiver problem. This raises the need for controlling the number of forwarders.

3.4 Load balancing problem

Opportunistic routing can conserve energy by reducing sender wait time. However, in order to achieve long network lifetime, it is also important to balance the traffic load among nodes in the network. ORW does not explicitly consider load balancing when computing forwarder sets. Thus, if network topology and link quality do not change, then forwarder sets do not change over time. Nodes with smaller EDC have higher chance of being selected as forwarders, so traffic load is concentrated at those nodes.

In order to balance traffic load among nodes, the forwarder selection metric should consider energy status of the nodes. Specifically, nodes with higher residual energy should be chosen as forwarders more frequently.

We propose a new metric called FS (Forwarder Score), which includes residual energy in the first term of EDC. The equation for calculating FS of node i is:

$$FS_i = \frac{1}{\zeta_i^{\alpha} \cdot (n_i + 1)} + \frac{\sum_{j \in F_i} FS_j}{n_i}, \tag{7}$$

where α is a system parameter. ζ_i can be calculated as:

$$\zeta_i = \left\lceil \frac{E_i}{E_0} \times g \right\rceil, \tag{8}$$

where g is the quantization granularity. For example, if g is 10, ζ_i is an integer value between 0 and 10. We assume that a node can estimate its remaining energy. Depending on the quantization granularity, the accuracy requirement for residual energy estimation will vary.

In Eq. (7), α determines how much weight is put on the residual energy. If α is zero, then FS falls back to the original simplified EDC. As α is increased, the impact of residual energy on FS becomes more significant. Since FS of a node is affected by FS of all possible upstream nodes, it will be relatively high if a node with low residual energy is included in the potential path from the node to the sink. Thus, nodes tend to avoid paths with low-energy nodes when selecting forwarders.

Forwarder selection using EDC is shown to create a DODAG structure and thus achieve loop-freeness. In Section 4, we show that FS also achieves the same property.

4 Proposed Protocol

The proposed protocol is called ORR (Opportunistic Routing based on Residual energy). Like ORW, each node maintains its forwarder set and the node forwards its packet to any of the nodes in the set. ORR has two major differences with ORW. First, ORR uses FS (Eq. (7)) for forwarder selection. Since the FS metric considers residual energy, a node with larger remaining energy has a higher chance of becoming other nodes' forwarder. Second, ORR controls the number of forwarders based on forwarding cost estimation. As shown in the performance evaluation, optimal number of forwarders varies based on network topology, traffic load, duty cycle, etc. ORR automatically calculates the number of forwarders that achieves near-optimal throughput regardless of network environment.

4.1 Deployment and management

Initial deployment and topology management of ORR is similar to that of ORW. Nodes periodically send HELLO messages to notify its existence to the neighbors. A HELLO message contains node ID, hop distance from the sink, and FS of the sender. Whenever a node discovers new neighbor or the hop distance information is updated, it sends a HELLO message to its neighbors so that the nodes can quickly adapt to the new topology. The hop distance information is used to establish a collection tree in the network. This structure is used for forwarding packets until nodes establish forwarder sets.

A node's forwarder set is initially empty, and FS is set to FS_0, which is a large value. After collecting neighbor information, a node can calculate its forwarder set based on the algorithm described in Section 4.2. According to Eq. (7), FS of a node depends on FS of all its neighbors. So a change in FS can potentially affect the forwarder set of all other nodes. If we let each node broadcast a HELLO message whenever its FS changes, the message cost will be too high. Instead, we use the same approach used in ORW: packets contain FS in the header, and neighbors may detect change of FS through overhearing and packet reception. Using this approach, the updated FS may not be propagated into the network quickly, but there is no additional message cost [27].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

ORR has another procedure for updating the forwarder set. As described in Section 4.3, the sink periodically calculates the optimal number of forwarders based on forwarding cost estimation. In the process, the sink computes the forwarder set of all nodes in the network. The updated forwarder set and FS of each node is distributed along with the optimal number of forwarders. So even though information propagates slowly without use of explicit messaging, nodes have a chance of updating forwarder set based on the newest information periodically.

In order to calculate n_{max}, the sink needs to gather information from the node. Specifically, the sink needs to know the current energy status and neighbor information of each node. The status information can be piggybacked in the data packets, or can be sent in a separate message if a node does not send any packet during the period. Due to the message cost, the period for calculating number of forwarders should not be too short. In the simulations, we set this period to be 24 hours.

4.2 Forwarder set calculation

As discussed previously, each node can calculate forwarder set based on neighbor information. Also, the sink periodically calculates forwarder sets for all nodes. In either case, the forwarder set is determined based on the FS metric (Eq. (7)). The forwarder set computation process is similar to ORW, except that the number of forwarders is limited by n_{max}, a parameter automatically calculated by the sink. Calculating n_{max} is described in the next section. Here we describe how node i computes its forwarder set.

If node i is a direct neighbor of the sink, its forwarder set includes only the sink node. Also, FS_i is calculated as:

$$FS_i = \frac{1}{\zeta_i^\alpha} \times \frac{T_a}{T}. \tag{9}$$

We assume that the sink is always active, and thus direct neighbors of the sink does not need to wait for the receiver. T_a/T is the duration of time needed to send a packet in unit of wake-up interval.

If node i is not a direct neighbor of the sink, Eq. (7) is used to determine its forwarder set. The procedure for establishing forwarder set is shown in Algorithm 1.

Algorithm 1 Forwarder selection process at node i

```
F_i = ∅, V = ∅, FS_i = FS_0
for each neighbor node j do
    insert j to V if FS_j < FS_i
end for
while V ≠ ∅ and |F_i| < n_max do
    remove node j with minimum FS from V
    if FS_j < FS_i then
        insert j to F_i
    else
        break
    end if
    recalculate FS_i
end while
```

When the sink computes forwarder set of all nodes, it follows the procedure described in Algorithm 2.

Algorithm 2 Procedure for computing forwarder set of nodes at the sink

```
U = ∅
for each node i that is a direct neighbor of the sink do
    calculate FS_i is according to Eq. (9)
end for
for each node i that is not a direct neighbor of the sink do
    F_i = ∅, FS_i = FS_0
    insert i into U
end for
while U ≠ ∅ do
    Dequeue a node i from U
    FS'_i ← FS_i
    Recalculate FS_i using Algorithm 1
    if FS_i < FS'_i then
        for each node j that is a neighbor of node i do
            if FS_i < FS_j then
                insert j into U
            end if
        end for
    end if
end while
```

It is important that the forwarder sets do not create loops in the network. To prove that the forwarding structure is loop-free, it is enough to say that FS of a node is always strictly larger than FS of every node in its forwarder set. In other words, upstream nodes always have smaller FS compared to downstream nodes. We prove this property in Lemma 1.

Lemma 1. $FS_c < FS_i, \ \ \forall c \in F_i$.

Proof. Consider the case where node i is building its forwarder set according to Algorithm 1. Suppose we come to a point where node i has k nodes in its forwarder set F_i. At this point, FS_i is:

$$FS_i = \frac{1}{\zeta_i^\alpha \cdot (k+1)} + \frac{\sum_{j \in F_i} FS_j}{k}. \tag{10}$$

Now, node i is considering of adding another node c to the forwarder set. Node c can be inserted to the forwarder set only when FS_c is smaller than FS_i. So the following inequality holds:

$$FS_i > FS_c. \tag{11}$$

Once node c is added to the forwarder set, the updated FS'_i is:

$$FS'_i \ = \ \frac{1}{\zeta_i^\alpha \ (k+2)} + \frac{\sum_{j \in F_i} FS_j + FS_c}{k+1} \tag{12}$$

The inequality between FS'_i and FS_c can be derived as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

$$
\begin{aligned}
& FS_i - FS_c \\
= \quad & \frac{1}{\zeta_i^{\alpha} \cdot (k+2)} + \frac{\sum_{j \in F_i} FS_j + FS_c}{k+1} - FS_c \\
= \quad & \frac{k}{k+1}\left(\frac{1}{\zeta_i^{\alpha}} \frac{k+1}{(k+2)\cdot k} + \frac{\sum_{j \in F_i} FS_j}{k} - FS_c\right) \\
> \quad & \frac{k}{k+1}\left(\frac{1}{\zeta_i^{\alpha} \cdot (k+1)} + \frac{\sum_{j \in F_i} FS_j}{k} - FS_c\right) \\
= \quad & \frac{k}{k+1}\left(FS_i - FS_c\right). \qquad (13)
\end{aligned}
$$

So whenever a new node c is inserted, the updated FS_i is larger than FS_c. Since nodes are inserted to F_i in the ascending order of FS, $FS_i < FS_c$ for all $c \in F_i$. □

4.3 Controlling number of forwarders

As previously discussed, allowing multiple forwarders have both positive and negative effect on the energy consumption. Thus, number of forwarders should be controlled considering both effects. In ORR, the sink periodically calculates n_{max}, a parameter used by all nodes in the network. As shown in Algorithm 1, a node stops inserting more nodes into the forwarder set when the number of forwarders reaches n_{max}.

The sink decides n_{max} based on forwarding cost estimation of the whole system. Considering the whole system is necessary because multiple receive problem occurred at a node not only extends wait time of that particular node, but increases energy consumption of upstream nodes by generating duplicate packets. To estimate forwarding cost, we need to answer the following to questions. First, what is the average cost (sender wait time) of a packet transmission? Second, what is the average number of transmissions, considering the duplicate packets? Here we find answers to these questions before calculating n_{max}.

4.3.1 Average sender wait time

To calculate average sender wait time, we use the slot-based model also used in Section 3.3.2. Suppose there are S slots in a wake-up interval. When a node transmits a packet, it continues transmission from the first slot to the S-th slot, or until when it receives an ACK. (We assume that the sender stops transmission after a wake-up period even if it does not receive an ACK.) Assuming the node has n forwarders we can calculate the probability that the transmission is successful (the sender receives an ACK) in the m-th slot ($p_{n,S}(m)$) using Lemma 2.

Lemma 2. *When there are S slots in a wake-up interval and the node has n forwarders, the probability that a successful transmission takes place in m-th slot ($p_{n,S}(m)$) is:*

$$p_{n,S}(m) = \sum_{k=1}^{\min(n,m)} (-1)^{k+1} \cdot {}_{m-1}C_{k-1} \cdot r_{n,S}(k),$$

where $r_{n,S}(k)$ defined as in Eq. (15).

Proof. Let us first consider the probability that transmission is successful in the first slot ($p_{n,S}(1)$). For this to happen, one of the n forwarders should wake up in the first slot, while all other nodes wake up at the later slots. Its probability is:

$$p_{n,S}(1) = n \cdot \frac{1}{S} \cdot \left(1 - \frac{1}{S}\right)^{n-1}. \qquad (14)$$

Now we consider the condition where the sender successfully sends its packet in the second slot. First, we calculate the probability that a single node wakes up in the second slot, and other nodes wake up in other slots (including the first slot). From this, we subtract the probability that a single node wakes up in the first and the second slot. To make the equations simple, we define r so that $r_{n,S}(k)$ is the probability that each of k nodes wakes up in the unique slots and other nodes wake up in other slots. For example, $r_{n,S}(2)$ is the probability that a node wakes up in the first slot and another node wakes up in the second slot, while other $n-2$ nodes wake up at other slots. The probability $r_{n,S}(k)$ can be calculated as:

$$r_{n,S}(k) = {}_nC_k \cdot k! \cdot \left(\frac{1}{S}\right)^k \left(1 - \frac{k}{S}\right)^{n-k} \qquad (15)$$

when $k \leq n$. If $k > n$, $r_{n,S}(k) = 0$. Using r, the probability $p_{n,S}(1)$ and $p_{n,S}(2)$ can be expressed as:

$$
\begin{aligned}
p_{n,S}(1) &= r_{n,S}(1) \\
p_{n,S}(2) &= r_{n,S}(1) - r_{n,S}(2). \qquad (16)
\end{aligned}
$$

Now we consider the probability that communication is successful in the m-th slot ($p_{n,S}(m)$). We start with the probability that a single node wakes up in the m-th slot, which is $r_{n,S}(1)$. From this, we subtract the probability that *a single node wakes up in the m-th slot, and also a single node wakes up in any of the slots from 1 to $m-1$*. Let us define A_i to be the event where a single node wakes up in slot i and also a single node wakes up in slot m. Then, we can write $p_{n,S}(m)$ as follows:

$$p_{n,S}(m) = r_{n,S}(1) - P(\cup A_i) \qquad 1 \leq i \leq m-1. \qquad (17)$$

Using the formula for probability of union of events, $P(\cup A_i)$ is:

$$
\begin{aligned}
P(\cup A_i) = \quad & \sum_i P(A_i) - \sum_{i<j} \sum P(A_i A_j) \\
& + \sum_{i<j<k} \sum \sum P(A_i A_j A_k) \\
& - \cdots + (-1)^m P(A_1 \cdots A_{m-1}) \qquad (18)
\end{aligned}
$$

In the considered scenario, $P(A_i)$ is equal for all i and it is $r_{n,S}(2)$. Similarly,

$$P(A_{i_1} \cdots A_{i_k}) = r_{n,S}(k+1). \qquad (19)$$

Thus,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

$$\begin{aligned} p_{n,S}(m) &= r_{n,S}(1) - \{ {}_{m-1}C_1 r_{n,S}(2) - {}_{m-1}C_2 r_{n,S}(3) \\ &\quad - \cdots + (-1)^m {}_{m-1}C_{m-1} r_{n,S}(m)\} \\ &= \sum_{i=1}^{m} (-1)^{i+1} \cdot {}_{m-1}C_{i-1} \cdot r_{n,S}(i). \end{aligned} \quad (20)$$

Since $r_{n,S}(i) = 0$ when $i > n$, this equation can be also written as follows:

$$p_{n,S}(m) = \sum_{i=1}^{\min(m,n)} (-1)^{i+1} \cdot {}_{m-1}C_{i-1} \cdot r_{n,S}(i). \quad (21)$$

□

Using $p_{n,S}(m)$, we can express the average sender wait time as:

$$t_{n,S} = \sum_{m=1}^{S} m \cdot p_{n,S}(m) / \sum_{m=1}^{S} p_{n,S}(m). \quad (22)$$

Fig. 6 shows average sender wait time obtained through analysis and simulation. It verifies the correctness of Eq. (22), and shows that as the number of forwarders increase, average sender wait time decreases up to some point, but stops decreasing and starts to increase when the number of forwarders is large.

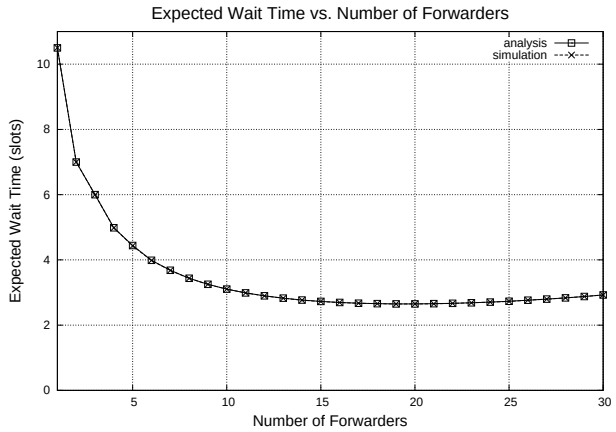


Fig. 6. Expected sender wait time varying number of forwarders. Number of slots per wake-up interval is 20.

4.3.2 Average number of packet transmissions

Now we calculate the average number of transmissions, considering duplicates generated by multiple receiver problem. Suppose the transmission is successful in the m-th slot. Among n forwarders, l nodes wake up before m-th slot, but the sender does not receive an ACK due to collision. In this case, although a single packet is transmitted, $l+1$ packets are generated as a result of this transmission. First, we define $\varphi_{n,S}(l,m)$, which is the probability that l nodes wake up before m-th slot, and a single node wakes up in the m-th slot. $\varphi_{n,S}(l,m)$ can be calculated as:

$$\varphi_{n,S}(l,m) = {}_nC_l \cdot {}_{n-l}C_1 \cdot \left(\frac{m-1}{S}\right)^l \left(\frac{1}{S}\right) \left(\frac{S-m}{S}\right)^{n-l-1}. \quad (23)$$

Next, we define $\vartheta_{l,m}$, the probability that transmission is not successful within m slots, even when l forwarders wake up during that time. $\vartheta_{l,m}$ can be calculated as:

$$\vartheta_{l,m} = 1 - \sum_{k=1}^{m} p_{l,m}(k). \quad (24)$$

Now, the probability $\varphi_{n,S}(l,m) \cdot \vartheta_{l,m-1}$ means that the transmission is successful at m-th slot, even though l forwarders wake up between first and (m-1)-th slot. Assuming 100% link quality, $l+1$ packets are generated after this transmission. We define $R_{n,S}$ as the expected number of received packets per transmission. It is:

$$R_{n,S} = \sum_{i=2}^{S} \sum_{j=1}^{l-1} ((j+1) \cdot \varphi_{n,S}(j,i) \vartheta_{j,i-1}) + n \cdot \vartheta_{n,S} \quad (25)$$

The last term accounts for the case where transmission is unsuccessful for the whole wake-up interval. Fig. 7 shows a plot of $R_{n,S}$ varying number of forwarders, obtained using analysis and simulation.

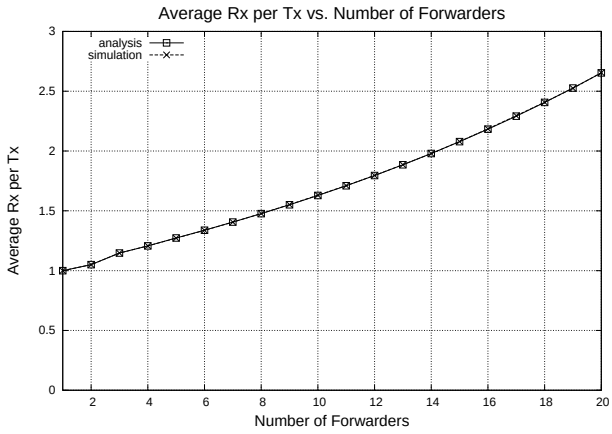


Fig. 7. Expected number of packets received per transmission. Number of slots per wake-up interval is 20.

4.3.3 Optimal number of forwarders

Now that we have calculated the expected wait time and the number of packets generated per transmission, we can estimate energy consumption of the nodes. Suppose a part of the network looks like the topology in Fig. 8. We want to find out the energy consumption of node A. First, we estimate how many packets node A will send. For that we need to consider A's downstream nodes. node A is in the forwarder set of node B and C. Also, B and C has two forwarders in the forwarder set each. In average half of B's packets will be received by node A, and so are the half of C's packets. Suppose node D is the only downstream node of both B and C. Half of D's packets will go to node B and the other half will go to node C.

If we assume that the point of packet generation is uniformly random, number of packets generated at each node will be approximately the same in all six nodes. Let us suppose for a certain duration of time, each node generates

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

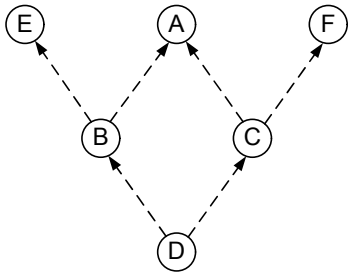


Fig. 8. An example topology. Energy consumption can be calculated from expected number of packet transmissions and average wait time.

one packet each. (We do not consider in-network aggregation in the analysis.) First, expected number of transmission at node D is 1. Expected number of packets at node B is:

$$Q_B = \frac{1}{n_D} \cdot R_{n_D,S} \cdot Q_D + 1. \quad (26)$$

Similarly, node A's expected number of transmissions can be calculated as:

$$Q_A = \sum_{i \in \widetilde{F_A}} \left(\frac{1}{n_i} \cdot R_{n_i,S} \cdot Q_i \right) + 1, \quad (27)$$

where $\widetilde{F_A}$ is the *reverse forwarder set* of A, which means that if node A is included in node i's forwarder set, i is in $\widetilde{F_A}$.

At each update period, the sink calculates the total energy cost for a range of number of forwarders. In other words, the sink tests if a certain n_{max} is the best for the network. The procedure is as follows. First, the sink node picks a value from the range. Using the value as n_{max}, it computes the forwarder set of all nodes using Algorithm 2. Then, it calculates the total energy cost using Eq. (27). After calculating energy cost for all candidate values, the one that has the minimum cost is chosen as n_{max}. For example, Fig. 9 shows an example forwarding cost calculated at the sink. In this case, the maximum number of forwarders is 11. Finally, the sink distributes the information to the nodes, along with calculated forwarder sets.

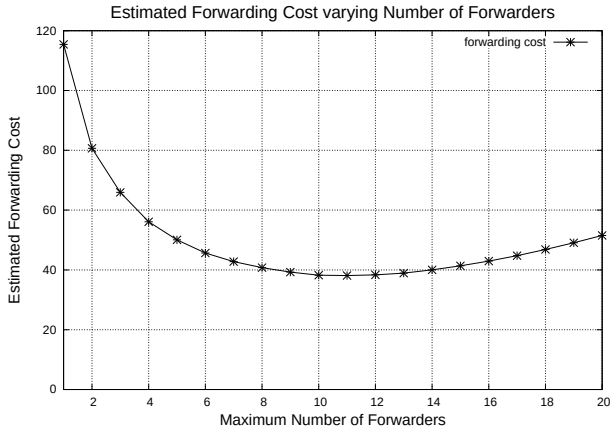


Fig. 9. An example cost estimation at the sink. Maximum number of forwarders becomes 11, which shows the minimum estimated forwarding cost.

To summarize, ORR controls number of forwarders based on forwarding cost estimation, and balances load by including residual energy in forwarder selection algorithm. It is important to note that although ORR can achieve higher network lifetime, both limiting number of forwarders and preferring nodes with higher residual energy results in longer packet latency. So if the application requirement is to achieve a certain level of packet latency, wake-up interval of ORR should be set shorter than wake-up interval of ORW. But as shown in the next section, ORR achieves significantly higher network lifetime compared to ORW even when their wake-up intervals are set differently to achieve the same average packet latency.

5 Performance Evaluation

5.1 Simulation Setup

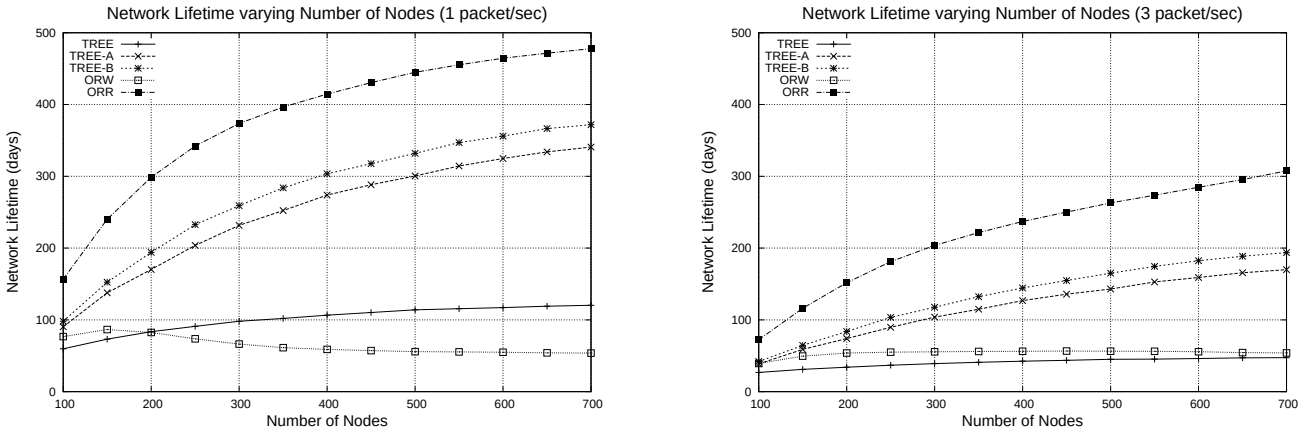
We have evaluated performance of ORR using a simulator written in C++. The simulator captures real-life events such as carrier sensing, backoff, and collisions. The sensor nodes run BoX-MAC [2], on top of which we implement various routing protocols and compare their performance, most significantly the network lifetime.

When the simulation begins, nodes are randomly placed in a 100m $\times$ 100m square-shaped area, and the sink is placed at the center. Transmission range of a node is 20m, and carrier sense range is 40m. This makes the maximum hop distance from the sink four to six. Also, the average number of neighbors is approximately 10% of the number of nodes. Once deployed, the nodes start duty-cycling, switching between active and sleep modes based on the pre-configured wake-up interval. Packets are generated at a random node according to the packet generation rate. When a packet is generated, the node immediately starts forwarding the packet towards the sink. We compare performance of the following protocols:

- TREE: TREE is a standard tree-based protocol, such as CTP [16]. Root of the tree is the sink, and each node has a single parent node where the packets are forwarded to.
- TREE-A: TREE-A is a tree-based protocol, but nodes change their parent nodes based on the residual energy.
- TREE-B: TREE-B is another variation of a tree-based protocol. Instead of considering parent's residual energy, a node considers residual energy of the *path* when selecting its parent node. The path energy is defined as the minimum residual energy of all nodes in the path to the sink.
- ORW: ORW is the protocol discussed in Section 3. Each node selects a set of forwarders based on the EDC metric. Link quality estimation is not included in the simulation as discussed in Section 3.2.
- ORR: ORR is the proposed protocol. It uses the FS metric when selecting the forwarder set. Periodically, the sink node calculates n_{max} and forwarder set of nodes, and distributes the information. This period is set to be 24 hours, and the cost of sending control messages is included in the results.

For all protocols, every node sends a HELLO message periodically. The HELLO message includes node information specific to protocols (residual energy for TREE-A, path

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing



(a) 1 packet/sec (b) 3 packets/sec

Fig. 10. Network lifetime varying number of nodes.

residual energy for TREE-B, EDC for ORW, and FS for ORR). When a node receives a HELLO message, it can change parents or recalculate forwarder sets if it detects an update on neighbor information. All data packets also carry node information in the header, so nodes can update neighbor information by receiving data packets.

The goal of the simulation is to evaluate how well the algorithms in ORR performs, in terms of network lifetime and packet delay. Specifically, the question we want to answer is: For a given network environment such as network density, traffic load, duty cycle, and link quality, what is the network lifetime and packet delay that each protocol will achieve? Other aspects of protocols, such as reacting to node mobility or topology change, are not covered. In the simulations, nodes do not move and link quality does not change over time. Although it is a limitation of the simulations, initial deployment and topology management schemes do not fundamentally differ between the protocols.

Every node (except the sink node which has unlimited energy) starts with 2000mAh, a typical capacity of an AA-type battery. Transmit mode, receive mode, and sleep mode current consumption are 17.4mA, 18.8mA and 0.02μA, respectively [1]. Active period is 50ms, and duration of receive check, invalid packet reception, and valid packet reception are 5.61ms, 20ms, and, 50ms, respectively, as specified in the BoX-MAC description [2]. The wake-up interval of each node is 1 second, unless otherwise specified. With 1 second wake-up interval and 50ms active period, the duty cycle is 5%. If a node has a packet to send but finds the carrier busy, it conducts a random backoff with an average of 30ms. After a node starts transmitting a packet, if it does not receive an ACK for 10 seconds, the node cancels the transmission and drops the packet. For TREE-A, TREE-B, and ORR, energy levels are quantized to 64 levels. For ORR, the default α is 2 and the period for calculating n_{max} is 24 hours. Other parameters such as number of nodes and packet generation rate are specified with the results. The simulation continues until one of the sensor nodes drains all of its energy. Network lifetime is defined as the duration of time until the first node dies out due to energy drain [28]. Each point in the graphs is an average of 1000 simulation runs with different topologies and random number generator seeds.

5.2 Results

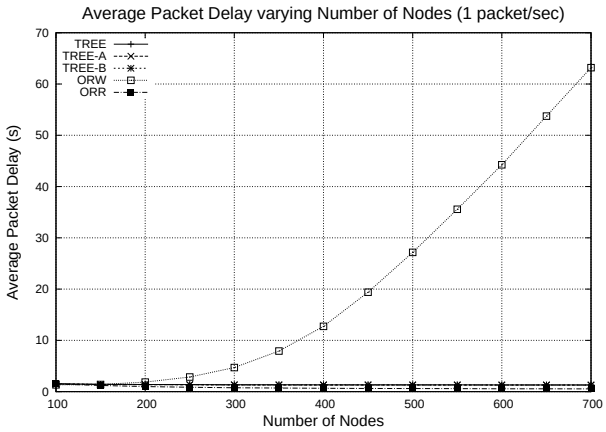
5.2.1 Varying number of nodes

In the first experiment, we varied the number of nodes from 100 to 700. With fixed area size, larger number of nodes means higher node density. Fig. 10(a) and (b) show the network lifetime when packet generation rate is 1 and 3 packets/sec, respectively. Several observations can be made from the results. First, network lifetime increases as node density becomes higher. Even without explicit load balancing, paths from sensor nodes to the sink are more diverged throughout the nodes when the density is higher. If the routing has a load balancing feature, such as TREE-A, TREE-B, and ORR, benefit of having large number of nodes becomes more significant, as can be seen in the graphs. Second, network lifetime is shorter with higher packet generation rate, which is obvious because more traffic leads to higher energy consumption. Third, network lifetime of ORW is comparable to TREE and shorter than other protocols. It is due to the negative effect of duplicate packets generated from multiple receiver problem. Because of this, performance of ORW is sensitive to duty cycle (5% in this case). Fourth, TREE-A and TREE-B protocols achieve longer network lifetime than TREE and ORW through load balancing, especially when node density is high. TREE-B slightly outperforms TREE-A by using path residual energy. Finally, ORR achieves the longest network lifetime regardless of network density and traffic load, a result achieved by controlling number of forwarders and load balancing. Compared with TREE-B, ORR achieves approximately 50% longer network lifetime.

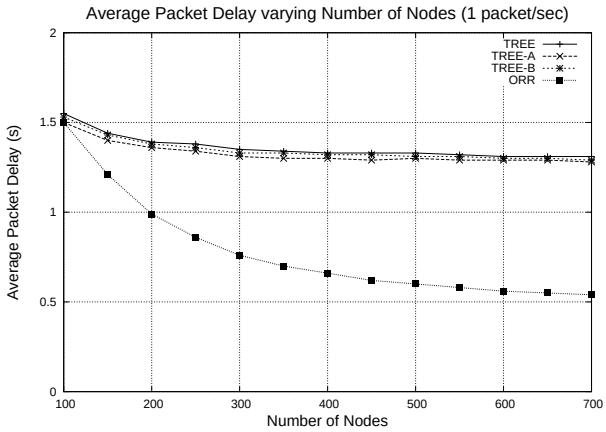
5.2.2 Metrics other than network lifetime

Although network lifetime is a very important factor characterizing performance of a wireless sensor network, there are other factors worth measuring. In this experiment, we have measured average packet delay, average energy consumption, average number of packet transmissions, and average residual energy of the second-hop nodes. In all experiments, packet generation rate is 1 packet per second and number of nodes is varied from 100 to 700. Fig. 11(a) and (b) show the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing



(a) ORW included



(b) ORW omitted

Fig. 11. Average packet delay varying number of nodes. Packet generation rate is 1 packet per second.

average packet delay. Packet delay is measured as the time between when a packet is generated and the time when the packet is received at the sink. The result in Fig. 11(a) shows that with increasing node density, packet delay of ORW increases significantly due to redundant forwarding. For visibility, Fig. 11(b) shows the result without ORW. All tree-based protocols have similar packet delays, because each node has a single forwarder. It can be seen that ORR achieves shorter packet delay through reduced sender wait time while successfully mitigating negative effects of redundant forwarding.

It was shown earlier that ORR achieves longer network lifetime compared to other protocols. However, it does not necessarily mean ORR spends less amount of energy, because network lifetime only considers time until the first node drains all of its energy. Some applications may be more interested in the total amount of energy consumed by the network. Fig. 12 shows the average energy consumption of a node in a day. Again, ORW consumes large amount of energy due to redundant forwarding, and energy consumption increases with node density. For the tree-based protocols, average energy consumption becomes smaller with more nodes, because traffic load stays similar while number of nodes increase. It can be observed that energy consumption of TREE, TREE-A, and TREE-B are similar. This result tells us that although TREE-A and TREE-B consume similar amount of energy as TREE, they achieve longer network lifetime through load balancing. ORR consumes approximately 15% less energy compared to TREE-B, but the difference in network lifetime is greater, as shown in Fig. 10. It is due to the fact that ORR achieves better load balancing compared to TREE-A and TREE-B (shown later).

Fig. 13 shows the average number of transmissions per packet. For all the tree-based protocols, it is the same as the average hop distance from the sink. ORR shows a slightly larger number of transmissions, whereas ORW requires a significantly large number of transmissions. The difference between two protocols is that ORR controls the number of candidate forwarders so that the number of redundant packets is kept small. Although ORR generates some redundant copies of packets, its energy consumption is lower than tree-based protocols thanks to shorter sender wait time.

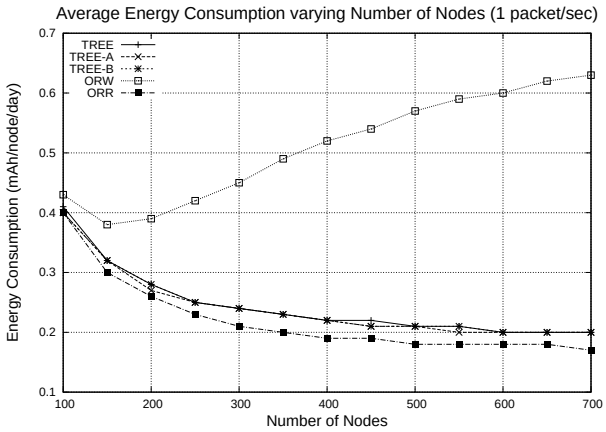


Fig. 12. Average energy consumption varying number of nodes.

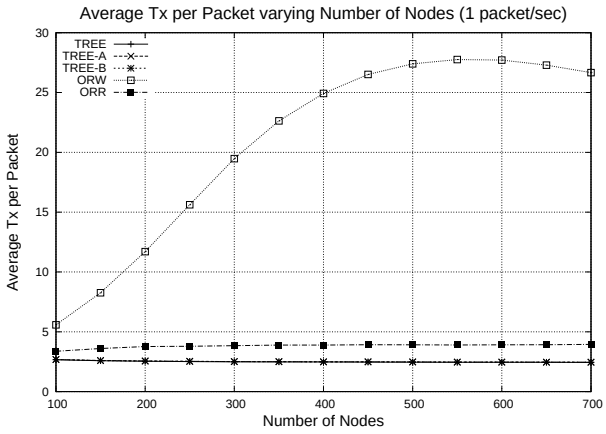


Fig. 13. Average number of Tx per packet varying number of nodes.

Fig. 14 shows the average residual energy of second-hop nodes from the sink, at the time when the first node drains all of its energy. This experiment is to study the performance of load balancing, because second-hop nodes are the ones that drain energy faster than other nodes. Direct neighbors

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

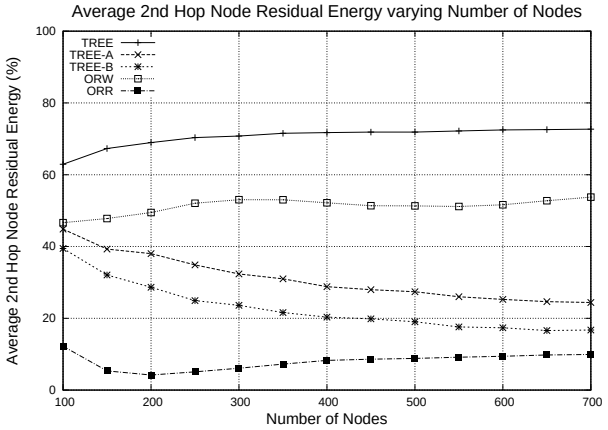


Fig. 14. Average residual energy of second-hop nodes.

of sink consume less energy, because the sink is always active and thus the first-hop nodes do not need to wait in active mode before transmission. If the average residual energy is low, it means that load is well-balanced among nodes. For the TREE protocol, the average residual energy is larger than 60%, showing the energy consumption is significantly unbalanced. TREE-A and TREE-B show similar patterns, in which the residual energy decreases when node density becomes higher. This pattern is due to the fact that when node density is higher, each node has more neighbors to choose its parent from. Thus, benefit of load balancing is greater with more nodes. TREE-B shows lower residual energy compared to TREE-A, indicating that using path residual energy can achieve better load balancing. ORR shows much less second-hop residual energy compared to ORW, because ORR considers load balancing when selecting forwarders. Also, the second-hop residual energy of ORR is significantly less than that of TREE-A and TREE-B. This means that ORR achieves better load balancing compared to TREE-A and TREE-B. It is because ORR establishes a DODAG, in which a node may choose a neighbor with larger hop distance from the sink as its forwarder. In tree-based protocols, on the other hand, a node must choose its parent from neighbors that have shorter hop distance from the sink.

5.2.3 Varying wake-up interval

As shown in Section 3.3.2, probability of multiple receiver problem depends on duty cycle and number of forwarders. In this experiment, we varied wake-up interval from 1 to 10 seconds to study the impact of duty cycle on protocol performance. Number of nodes is 300 and 1 packet is generated every second. The results on network lifetime and average packet delay are shown in Fig. 15.

For ORW and ORR, network lifetime increases with wake-up interval to a certain point, and then starts to decrease. When the wake-up interval is small, energy consumption from periodic wake-ups is larger than energy consumption from packet transmission. Thus, higher network lifetime can be achieved by increasing wake-up interval. However, when the wake-up interval becomes large, sender wait time becomes a dominant source of energy consumption. Since larger wake-up interval leads to longer sender wait time, network lifetime decreases with increasing wake-up interval. For the tree-based protocols, sender wait time is a dominant factor even when the wake-up interval is 1 second, since nodes only have a single forwarder. For opportunistic protocols, redundant packets generated from multiple receiver problem is another source of energy drain. Thus, ORW achieves shorter network lifetime than tree-based protocols when the wake-up interval is small, as shown in Fig. 15(a). Although ORR generates redundant packets as well, the energy consumption due to redundant forwarding is kept small by controlling number of forwarders. Since wake-up interval is not automatically controlled, network operator may not select the wake-up interval that achieves the highest network lifetime. Still, ORR achieves higher network lifetime compared to other protocols in a reasonable range of wake-up intervals.

Fig. 15(b) shows the average packet delay. ORW and ORR achieve shorter packet delay compared to tree-based protocols by allowing multiple forwarders. Also, packet delay of ORR is higher than that of ORW. That is because the two techniques used in ORR – limiting number of forwarders and preferring paths with higher residual energy – both lead to higher packet delay. So if the application requires low delay, ORR should use a shorter wake-up interval compared to ORW in order to meet the requirement. Nevertheless, when the wake-up intervals are selected to achieve the same delay goal, ORR achieves higher network lifetime than ORW.

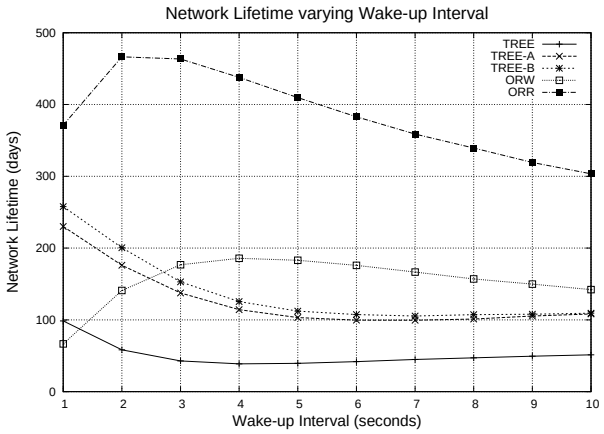
5.2.4 Impact of parameter α

In this experiment, we varied the parameter α in the FS equation (Eq. (7)) from 0 to 4. When α is 0, FS does not consider residual energy at all. When α is large, nodes with large amount of residual energy will have very low single-hop FS. In contrast, nodes with small amount of energy will have relatively high single-hop FS, so their chance of getting selected as a forwarder becomes low. In other words, nodes avoid choosing a path that includes low-energy nodes. Fig. 16 shows the network lifetime and average packet delay when number of nodes is varied from 100 to 700. Packet generation rate is 1 packet per second. The network achieves longer lifetime when α is higher, but the difference is minimal when α is larger than 1.0. That is because although large α achieves better load balancing, it creates longer paths in order to avoid low-energy node. The cost and benefit cancel out each other when α is large. Creating longer paths is also reflected in Fig. 16(b), where it is shown that larger α leads to longer average packet delay.

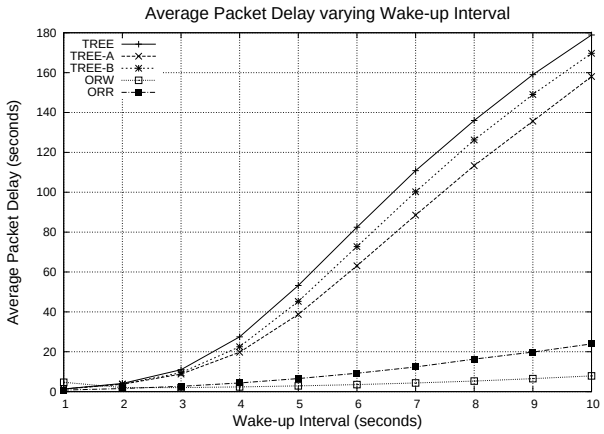
5.2.5 Performance of parameter auto-tuning

Automatically tuning forwarder set size is one of the main features in ORR. In this experiment, we study the performance of the auto-tuning algorithm by comparing with fixed forwarder set sizes. In Fig. 17, we varied the maximum number of forwarders (n_{max}) from 1 to 20 and measured network lifetime. The three curved lines represent wake-up interval of 1.0, 2.0, and 4.0 seconds respectively. The three straight lines are the results when auto-tuning is used. First, it can be observed that the optimal number of forwarders varies depending on the wake-up interval, which is intuitive because multiple receiver probability depends on the duty

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing

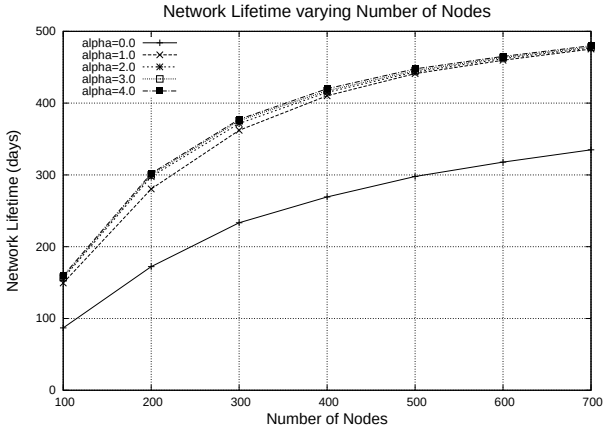


(a) Network lifetime

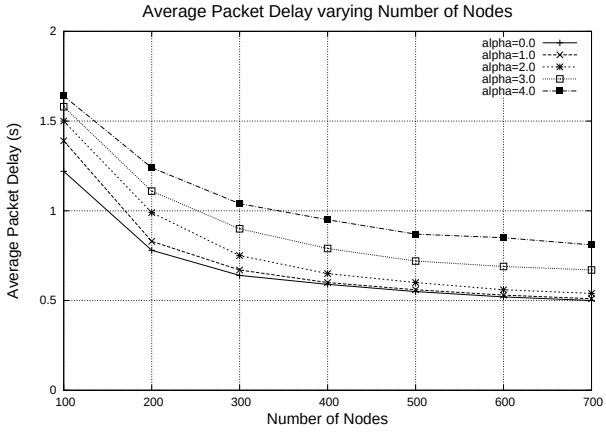


(b) Average packet delay

Fig. 15. Network lifetime and average packet delay varying wake-up interval. Number of nodes is 300 and 1 packet is generated every second.



(a) Network lifetime



(b) Average packet delay

Fig. 16. Network lifetime and average packet delay varying number of nodes. Packet generation rate is 1 packet per second.

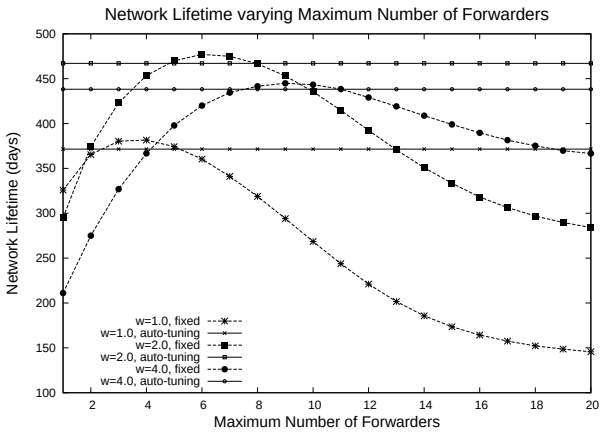


Fig. 17. Network lifetime varying number of forwarders.

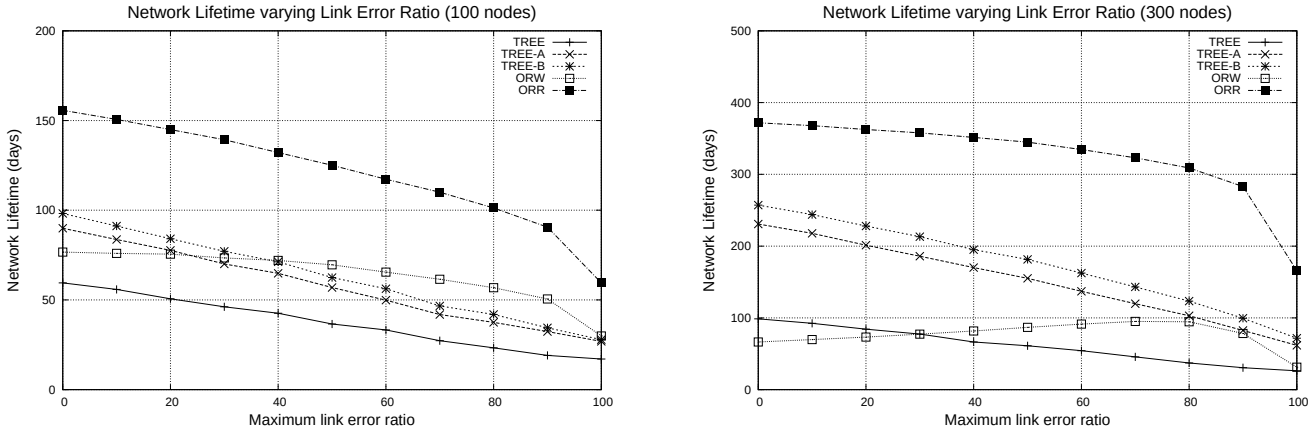
cycle. As shown in the graph, when wake-up interval is 1 second, the optimal number of forwarders is approximately 4. On the other hand, when the wake-up interval is 4 seconds, the optimal number of forwarders is approximately 10. By using parameter auto-tuning, we were able to achieve near-optimal network lifetime regardless of the wake-up interval. In all three cases, network lifetime achieved by parameter auto-tuning was over 97% of the longest network lifetime achieved using a fixed forwarder set size. In other words, the algorithm based on the forwarding cost estimation successfully finds a good n_{max} regardless of the environment.

5.2.6 *Impact of link quality*

Until now, link quality was set to 100% for all links. This means that when sender and receiver are located within the transmission range, the receiver will always receive the sender's packet unless there is a collision. In practice, packets can be lost due to fading or external interference. In this experiment, we study the impact of packet loss on protocol performance.

To model packet loss, each link is assigned an error ratio. When a packet is received at the node, whether the packet is received correctly is determined probabilistically using this ratio. The error ratio of each link is chosen randomly from 0 to e_{max}, the maximum error ratio. e_{max} is the parameter varied in the experiment. This model captures the real-life environment where channel conditions vary among links.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2606427, IEEE Transactions on Mobile Computing



(a) 100 nodes (b) 300 nodes

Fig. 18. Network lifetime varying maximum link error ratio e_{max}. Each link is assigned an error ratio between 0 and e_{max}.

The experiment is done for 100 and 300 nodes to see the results for networks with different density. The result is shown in Fig. 18.

When the link error ratio increases, network lifetime of all protocols are shortened. This is intuitive, because the sender stays longer in the active mode trying to deliver its packet and get an ACK. But ORW shows a different pattern compared to other protocols. In the 100-node scenario, the speed at which network lifetime decreases is much slower for ORW than other protocols. In the 300-node scenario, network lifetime of ORW even increases with the link error ratio, until the error ratio becomes significantly high. This is an indication that ORW is generating too many duplicate packets; energy which should have been wasted by duplicate packets is saved by packet losses. For ORR, number of forwarders is already controlled for minimum energy consumption. So increased sender wait time caused from packet losses shortens the network lifetime. Nevertheless, ORR achieves higher network lifetime compared to ORW regardless of the link quality.

6 Discussion

When designing ORR, we did not include the link quality term in the forwarder selection metric due to the inaccuracy of estimation scheme, as discussed in Section 3.2. However, packet losses can occur in real life due to various reasons, and some links may have a very low reliability. Since the metric FS assumes all links have equal quality, variation in link quality can can lead to error in forwarding cost estimation and degrade performance. When opportunistic routing is used, nodes have low probability of overhearing packets from neighbors. Thus, nodes have to explicitly send probe messages to measure link quality, which is very costly. One possible technique for removing unreliable links is to count the number of packets forwarded to its forwarders. If one of the forwarders shows a low packet count, it can be suspected that the link between sender and the forwarder is unreliable. Based on a threshold, the node can be removed from the neighbor list. This scheme does not rely on overhearing and does not require explicit messaging, although it does not accurately estimate the quality of each links.

In ORW [3], three "light-weight" techniques are described that can mitigate the multiple receiver problem: probabilistic ACK, overhearing, and network-layer duplicate detection. We briefly discuss these techniques here. In probabilistic ACK, nodes receiving a packet waits for a random delay before transmitting an ACK. This is to let the sender successfully receive multiple ACKs when there are multiple receivers. If the sender receives multiple ACKs, it retransmits the packet. If the previous receivers receive the retransmitted packet, the nodes send ACK with reduced probability in order to avoid collision. This scheme may not be useful for energy conservation due to the following reasons. First, if the sender receives multiple ACKs, it means that already more than one node have received the packet. Retransmitting the packet consumes more energy unnecessarily. Moreover, since there are other forwarders in the forwarder set, a retransmitted packet may be received not by the previous receivers but by other forwarders. This will further generate duplicate packets. Overhearing and network layer duplicate detection are opportunistic approaches that can remove some duplicates from the network. If a node overhears a packet and finds out that it has the same copy, the node drops the packet. Also, if a node happens to receive multiple copies of the same packet, the node only forwards the first one and drops the other packets. These two schemes are not harmful to the performance, and can be applied to both ORW and ORR. However, they have a limited effect because overhearing probability is low (Section 3.2), and network-layer duplicate detection can take place only when two copies of the same packet are delivered to the same forwarder on their path to the sink.

One thing to note is that even without injecting random delay between packet reception and ACK transmission, the packet sender may still receive multiple ACKs successfully. This can happen due to variance in interrupt handling at the receivers. Even if two nodes receive a packet at the same time, they may transmit ACKs at slightly different times due to variance in processing delay. If the difference is large enough so that the packets do not overlap with each other, the sender can receive both ACKs. If the node receives multiple ACKs, the node will stop transmitting the packet

at the first ACK and go back to sleep, although the duplicate packets received by the receivers would still be forwarded towards the sink. At least, the sender will spend less energy and will not generate further duplicate packets at other nodes. In our evaluation, it was assumed that the ACKs will always collide when multiple receivers receive the same packet. Due to this reason, the actual network lifetime could be longer than the results obtained in the simulations. The benefit of successfully receiving multiple ACKs will be greater for ORW than ORR, because ORW suffers more from multiple receiver problem. Still, although the possibility of receiving multiple ACKs will reduce energy waste caused from multiple receiver problem, it creates duplicates in the network which is harmful for network lifetime.

Calculating n_{max} based on forwarding cost estimation is one of the main features in ORR. The sink decides n_{max} that is applied to all nodes in the network. The reason that n_{max} is computed at the sink and not at individual nodes is because forwarding cost of a node is affected by all other nodes in the network. The forwarding cost depends on two things: the forwarder set and the expected number of packet transmissions. The forwarder set is affected by upstream nodes, whereas the expected number of packet transmissions depends on downstream nodes. So it is difficult to design a greedy-style heuristic to calculate n_{max} in a distributed manner. The sink tries different values of n_{max} to find one with the minimum forwarding cost. It is possible to allow different n_{max} for each nodes, but the computational complexity will be dramatically increased. A distributed algorithm for finding the optimal number of forwarders at each node will be pursued as a future work.

7 Conclusion

Opportunistic routing protocols particularly work well with asynchronous MAC protocols where sender has to wait until its receiver wakes up. By allowing multiple candidate receivers, sender wait time which is the major source of energy consumption can be reduced. However, the problem where multiple receivers wake up simultaneously and receive packets causes redundant forwarding, which severely degrades energy efficiency in the network. Also, lack of load balancing leads to some nodes draining their energy faster than others, losing coverage and connectivity. The proposed protocol ORR addresses these problems by including residual energy factor in the forwarder selection algorithm, and controlling number of forwarders based on forwarding cost estimation. Results from extensive simulations show that ORR can truly achieve benefit of opportunistic routing by mitigating the negative effects caused by redundant packet forwarding.

References

[1] C. AS, *SmartRF CC2420 Preliminary Datasheet (rev 1.2)*, 2004.

[2] D. Moss and P. Levis, "Box-macs: Exploiting physical and link layer boundaries in low-power networking," Stanford University, Tech. Rep. 08-00, 2008.

[3] O. Landsiedel, E. Ghadmi, S. Duquenoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *ACM/IEEE IPSN*, 2012, pp. 185–196.

[4] A. Bachir, M. Dohler, T. Watteyne, and K. Leung, "Mac essentials for wireless sensor networks," *IEEE Communication Surveys and Tutorials*, vol. 12, no. 2, pp. 222–248, 2010.

[5] W. Ye, J. Heidemann, and D. Estrin, "An energy efficient mac protocol for wireless sensor networks," in *IEEE INFOCOM*, 2002, pp. 1567–1576.

[6] P. Lin, C. Qiao, and X. Wang, "Medium access control with a dynamic duty cycle for sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2004, pp. 1534–1539.

[7] C. Merlin and W. Heinzelman, "Duty cycle control for low-power-listening mac protocols," *IEEE Transactions on Mobile Computing*, vol. 9, no. 11, pp. 1508–1521, 2010.

[8] Y. Sun, S. Du, O. Gurewitz, and D. Johnson, "Dw-mac: A low latency, energy efficient demand wakeup mac protocol for wireless sensor networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2008, pp. 53–62.

[9] Y. Zhao, C. Miao, and M. Ma, "An energy efficient self-adaptive duty cycle mac protocol for traffic-dynamic wireless sensor networks," *Wireless Personal Communications*, vol. 68, no. 4, pp. 1287–1315, 2012.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[11] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," in *ACM SenSys*, 2006, pp. 307–320.

[12] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," *Ambient Intelligence*, pp. 115–148, 2005.

[13] M. Miller and N. Vaidya, "A mac protocol to reduce sensor network energy consumption using a wakeup radio," *IEEE Transactions on Mobile Computing*, vol. 4, no. 3, pp. 228–242, 2005.

[14] A. Bachir, D. Barthel, M. Heusse, and A. Duda, "Micro-frame preamble mac for multihop wireless sensor networks," in *IEEE Int'l Conference on Communications (ICC)*, 2006, pp. 3365–3370.

[15] T. Park, K. Park, and M. Lee, "Design and analysis of asynchronous wakeup for wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 11, pp. 5530–5541, 2009.

[16] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *ACM SenSys*, 2009, pp. 1–14.

[17] S. Biswas and R. Morris, "Exor: Opportunistic multihop routing for wireless networks," in *ACM SIGCOMM*, 2005, pp. 133–144.

[18] M. Zorzi and R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Multihop performance," *IEEE Transactions on Mobile Computing*, pp. 337–348, 2003.

[19] L. Cheng, J. Niu, J. Cao, S. Das, and Y. Gu, "Qos aware geographic opportunistic routing in wireless sensor networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1864–1875, 2014.

[20] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable links," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2007, pp. 321–334.

[21] D. Liu, Z. Cao, J. Wang, Y. He, M. Hou, and Y. Liu, "Duplicate detectable opportunistic forwarding in duty-cycled wireless sensor networks," in *IEEE ICNP*, 2013, pp. 1–10.

[22] J. So and H. Byun, "Opportunistic routing with in-network aggregation for asynchronous duty-cycled wireless sensor networks," *Springer Wireless Networks*, vol. 20, no. 5, pp. 833–846, 2014.

[23] D. Singh, P. Kuila, and P. Jana, "A distributed energy efficient and energy balanced routing algorithm for wireless sensor networks," in *International Conference on Advances in Computing, Communication and Informatics (ICACCI)*, 2014, pp. 1657–1663.

[24] Y. Liao, H. Qi, and W. Li, "Load-balanced clustering algorithm with distributed self-organization for wireless sensor networks," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 1498–1506, 2013.

[25] Z. Hong, R. Wang, and X. Li, "A clustering-tree topology control based on the energy forecast for heterogeneous wireless sensor networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 1, pp. 68–77, 2016.

[26] S. Zairi, B. Zouari, E. Niel, and E. Dumitrescu, "Node self-scheduling approach for maximizing wireless sensor network lifetime based on remaining energy," *IET Wireless Sensor Systems*, vol. 2, no. 1, pp. 52–62, 2012.

[27] E. Ghadimi, O. Landsiedel, P. Soldati, S. Duquennoy, and M. Johansson, "Opportunistic routing in low duty-cycle wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 4, p. 67, 2014.

[28] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, 2009, article no. 5.