2011 IEEE 17th International Conference on Parallel and Distributed Systems

Dynamic Data Forwarding in Low-Duty-Cycle Sensor Networks

Yi Duan[1], Xiaobing Wu[1], Fan Wu[2], and Guihai Chen[1,2]
[1]State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing, 210093, China
[2]Shanghai Key Laboratory of Scalable Computing and Systems
Shanghai Jiao Tong University, Shanghai 200240, China
csduanyi@hotmail.com, wuxb@nju.edu.cn, fwu@cs.sjtu.edu.cn, gchen@nju.edu.cn

***Abstract*—In wireless sensor networks (WSNs), asynchronous duty-cycle technique can significantly reduce energy consumption. However, packets in low-duty-cycle networks suffer high end-to-end (E2E) delay. Besides, recent experimental studies have also shown that links in WSNs are highly unreliable and radio irregularity has adverse impact on routing protocols. In this work, we introduce a dynamic data forwarding (DDF) scheme which combines a realistic link model with asynchronous duty cycle. Different from most of other routing protocols, each node in our solution first finds out a set of candidate nodes and then forwards packet to the first waking up node in this set. Our solution can reduce E2E delay, guarantee delivery ratio and improve network lifetime. We evaluate this dynamic data forwarding scheme with extensive simulations and the simulation results demonstrate the efficiency of our solution.**

***Index Terms*—Asynchronous, duty cycle, dynamic data forwarding, realistic link model.**

I. Introduction

Wireless sensor networks can be deployed for various applications. In most of these applications, sensor nodes are powered by energy-limited batteries and it is infeasible to change batteries in some applications (*e.g.* environment monitoring in untraversed place). In order to maintain longer operation, it is important to save energy for sensor nodes while operating. Duty cycle [1], [2], [3], [4] is such a mechanism to bridge the gap between limited energy supplies and network lifetime. In this technique, each sensor node turns its radio on for a short time and then stays dormant for a long time, alternating between active and sleeping states. A lot of MAC protocols based on this technique have been proposed. There are two categories of duty-cycle MAC protocols [1], [2], [3], [4], synchronous and asynchronous. In synchronous approaches, neighboring nodes synchronize their clocks and have the same duty-cycle schedule. Asynchronous approaches on the other hand, allow nodes to operate independently, with each node on its own duty-cycle schedule. In this paper we focus on routing problem over WSNs working on asynchronous duty-cycle schedule, because asynchronous duty cycle is easy to implement, consumes no energy required for synchronizing and achieves excellent idle energy savings.

Routing in asynchronous duty-cycle WSNs results in a time-varying latency, because the relay node discovery time is varied. Packets under traditional routing protocols such as ETX [13], may suffer high E2E delay in multi-hop duty-cycle WSNs, since the next-hop node is predetermined. For example, when the source node has data to transmit, it has to wait for the predetermined relay node to wake up even though there are nodes waking up earlier than the predetermined node. If we choose the node that first wakes up as relay node, the E2E delay can be greatly decreased, especially in low-duty-cycle WSNs. However, nodes under this intuitional forwarding method will consume much energy if the link quality between source node and relay node is poor. So we should make a tradeoff between delay and energy consumption.

Recent experimental studies have revealed quite a few interesting results. Zhou *et al.* [11] show that radio irregularity on WSNs has adverse impact on protocols, especially on routing protocols. The most important result is that radio is non-isotropic, *i.e.* radio signal from a transmitter has different path loss in different directions. Zuniga *et al.* [12] give a detailed analysis of the transitional region in WSNs. In transitional region, link quality is highly dynamic. So, a good link metric that can reflect this irregularity is essential in routing protocols.

In this paper we propose a dynamic data forwarding (DDF) scheme which applies a realistic radio model to calculate link quality and combines it with duty cycle technique. DDF can achieve high delivery ratio, low E2E delay and prolong network lifetime.

The rest of this paper is organized as follows. In Section II, we review the related work. Section III describes the system model. The detailed design of our protocol is given in Section IV. The protocol is evaluated by simulation in Section V. We conclude this paper in Section VI.

II. Related Work

Routing in WSNs has attracted extensive attention in recent years. Different applications need different routing protocols since WSNs have special characters which are quite different from traditional wired and wireless networks. No routing protocol can be applied to all applications, so lots of routing protocols have been proposed.

Routing paths in some routing protocols are predetermined and are updated periodically or only when network topology changes. De Couto *et al.* [13] propose a metric called expected transmission count (ETX) to find out high throughput paths.

1521-9097/11 $26.00 © 2011 IEEE
DOI 10.1109/ICPADS.2011.53

But ETX suffers high E2E delay in low-duty-cycle WSNs. Karp *et al.* [5] introduce a geographic routing (GPSR) using geographic coordinates. Fonseca *et al.* [6] present another geographic routing using virtual coordinates. It does not need localization hardware (*e.g.* GPS) or localization algorithms. In geographic routing, packets are always forwarded to the neighbor which is closest to the destination (*e.g.* sink). However, this geographic greedy routing is not efficient since it may choose poor link quality node as relay.

All these routing approaches suffer high E2E delay in low-duty-cycle WSNs. Moreover, routing paths in these routing protocols are updated periodically or only when network topology changes. So they can not reflect the link quality change efficiently, especially in highly dynamic networks.

In other forwarding approaches, data forwarding path is time varying. On-demand routing protocols, such as AODV [7], and DSR [8], use broadcasting to find routing paths. These approaches consume much energy in routing discovery. Gu *et al.* [14] introduce a dynamic switch-based forwarding (DSF) scheme, which combines delivery ratio, E2E delay and energy consumption together. Though it shows good performance, it consumes much energy in synchronization since it works under synchronous duty-cycle schedule and the initialization and update cost is also high. Anycast [9], [10] technique is another kind of dynamic routing schemes which is proposed specially for WSNs working on duty cycle. In anycast technique, each sensor node forwards packet to the first node that wakes up among candidate next-hop nodes. However, the first waken up node might be a poor link quality node.

We note that a good link quality metric such as the one proposed in [12] is necessary to reflect the highly dynamic link quality. Choosing a high link quality node as relay node can achieve high network performance (*e.g.* throughput). On the other hand, to reduce E2E delay is also important in low-duty-cycle WSNs. However, these two goals may not be reached at the same time. Therefore, it is critical to make a tradeoff between choosing optimal link quality relay node and reducing E2E delay.

III. System Model

In this section we introduce the asynchronous duty-cycle network model and a realistic link model. We will give the routing protocol based on these models in next section.

A. Network Model

DDF is a dynamic routing protocol designed for low asynchronous duty-cycle WSNs. We assume that network has N sensor nodes, each sensor node has two states at a given time: active and dormant. When a node is in active state, it can sense the environment, transmit and receive packets. When a node is in dormant state, it turns all its function modules off except a timer to wake itself up. A dormant node wakes up when timer is expired or it has packets to transmit, but node can receive packets only when it is in active state. Nodes decide their schedules independently, and they do not synchronize their clocks. In short, nodes work asynchronously and can transmit at anytime but receive only when they are in active state. We give a simple example about asynchronous duty cycle in Fig. 1 where node A and node B have different working schedules. Note that the neighbor discovery latency is time varying since nodes have different schedules.

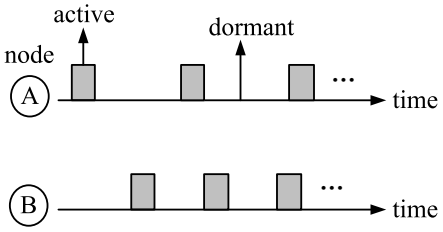


Fig. 1. Asynchronous duty cycle

B. Link Model

Here we introduce the link model proposed in [12]. For MICA2 node using the Chipcon CC1000 radio, the packet reception rate (PRR) is given as follows:

$$PRR = (1 - \frac{1}{2}\exp^{-\frac{\gamma}{2*0.64}})^{8\rho f} \quad (1)$$

where γ is the SNR (signal-to-noise ratio), ρ is the encoding ratio and f is the frame length. However we can not get SNR but RSSI (received signal strength indicator) from the mote. If we know the noise floor and RSSI, we can use them to calculate SNR. Since the noise floor is hard to determine, we can simply assume the noise floor is constant (*e.g.* -105dBm given in [12]).

Experiment studies have shown that links may be asymmetric [11], [16], so we can denote the link quality between two neighbor nodes as the sum of bidirectional ETX (expected transmission count). Single direction ETX can be calculated as follows:

$$ETX = \frac{1}{PRR} \quad (2)$$

On the operating of a network, we use RSSI to calculate ETX. For example, node i has to transmit a packet to a neighbor node j, when node j receives this packet, it reads the RSSI from radio with the received packet and replies an ACK message with the RSSI it has read. Here we denote the RSSI as $RSSI(i,j)$ which means this RSSI is read with a packet from node i to node j. Once node i receives this ACK, it can read the RSSI about this ACK (denoted as $RSSI(j,i)$) from radio and read the $RSSI(i,j)$ from ACK packet. With these two RSSIs, node i can calculate the link quality between node i and node j. We denote the link quality between node i and node j as follows:

$$LQ(i,j) = ETX(i,j) + ETX(j,i) \quad (3)$$

where $ETX(i,j)$ is the expected transmission count from node i to node j. Therefore, the smaller LQ is, the better the link quality is. In order to reflect the changes of network, we also propose a scheme to update LQ. We will introduce this scheme in section IV-B.

IV. DDF Protocol Design

In this section, We give a novel data forwarding scheme based on the models given in previous section. The main idea of DDF is shown in Fig.2. Here, node S is the source node and it has to transmit packets to the sink via its neighbors. All the nodes in the circle are node S's neighbors and the numerals are the waking up orders of these nodes. If node S chooses the first waking up node A as the relay node (Anycast case), it might consume much energy since the link quality via node A to the sink is poor. If node S chooses the best link quality node C as relay node (static routing protocol case), it has to wait a little more time for node C to wake up, this might increase the E2E delay. So our solution is to select several candidate nodes which have relative good link quality to the sink and transmit packet to these candidates one by one according to their waking up time until certain one candidate receives this packet, our solution makes a trade-off between energy consumption and E2E delay. In Fig.2, node S will choose node B, C and D as candidate nodes and transmit packet to node B,C,D according to their waking up time until one of them receives this packet.

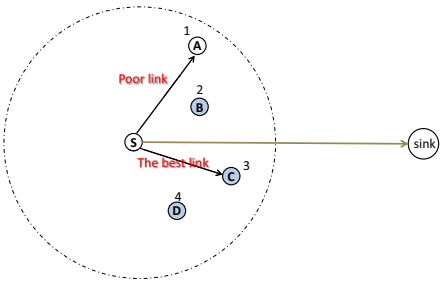


Fig. 2. Main idea of DDF

In order to present a deep insight into the data forwarding process, we give a brief introduction of X-MAC [4]. When a node has data to transmit, it broadcasts short preambles with target addresses (*e.g.* ID number of node) continuously until it receives an early acknowledgement packet during the short pause between two short preambles. After the sender receives the ACK from the target node, it starts to send out data packet. Fig.3 gives an example of the transmission process from node A to node B.

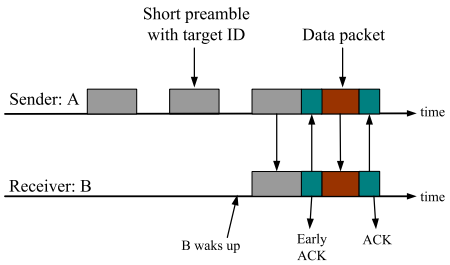


Fig. 3. Transmission process (X-MAC)

In our protocol, each node first sets up a neighbor table, then node uses this table to forward packets. Each node also updates its neighbor table after successfully forwarding a packet to its neighbor. Our protocol also solves the problems of retransmission and routing loops. We give detailed design of our protocol in following subsections.

A. Forwarding Strategy

In DDF, all sensor nodes maintain a table of its neighbors (Table I offers an example of such a table), and sensors use this table to forward packets. Each tuple of the table has two items, one is the ID number of neighbor, and the other is $< LQ, h >$ where LQ is the link quality to the sink (*see below*) via this neighbor node, and h is the hop distance from neighbor node to the sink. The table is also sorted in descending order relative to $< LQ, h >$ and Definition 2 shows how to compare two $< LQ, h >$s. We'll present how to set up this table in section IV-B in detail.

TABLE I
NEIGHBOR TABLE

neighbor	$< LQ, h >$
N_1	$< 6.6560,\ 3 >$
N_2	$< 7.5153,\ 3 >$
N_3	$< 9.0758,\ 4 >$
...	...

Definition 1 (Link Quality to Sink(LQ)): $LQ_i(j)$ is the link quality from node i to the sink via node j and can be calculated as follows:

$$\begin{cases} LQ_i(j) &= LQ(i,j) + \widehat{LQ_j} \\ \widehat{LQ_j} &= \min_{k \in N(j) \setminus \{i\}}(LQ_j(k)) \end{cases} \tag{4}$$

where $LQ(i,j)$ is the link quality from node i to node j, $\widehat{LQ_j}$ is the best link quality to the sink that node j can achieve via its neighbor except node i, and $N(j)$ is the set of node j's neighbors whose hop-count is no more than node j's one.

Definition 2: We say $< LQ_i, h_i > \ > \ < LQ_j, h_j >$, if $LQ_i < LQ_j$ or $LQ_i = LQ_j$ *and* $h_i < h_j$. $< LQ_i, h_i > = < LQ_j, h_j >$, if and only if $LQ_i = LQ_j$ *and* $h_i = h_j$. The Definition 2 conforms with our intuition, *i.e.* we prefer higher link quality node as relay, and when LQ_i equals LQ_j, we prefer the shorter path to forward data.

When a node i has data to transmit, it first queries the table and finds out the candidate nodes via which node i has a better LQ to the sink than a threshold θ, *i.e.* we choose node whose LQ is smaller than θ. Note that θ is not a constant but a function of hop-count and α. The function is given as follows:

$$\theta = 2h * \frac{1}{\alpha} \tag{5}$$

where α is the lower bound of PRR we expect, h is the hop-count from node i to the sink, and the factor 2 means that we consider bidirectional link quality. The larger lower bound α is, the smaller the threshold is and the less candidate nodes will be chosen and vice versa. When a node finds out the candidate

nodes, it puts the IDs of these candidates in the preamble and starts the data forwarding process.

Note that only next-hop and the same hop nodes can help deliver packets towards the sink, so neighbor table only records these nodes. We also note that some node may find no candidates according to the threshold θ and retransmission is important to achieve high delivery ratio. We will solve these problems in section IV-C. We find that multi-candidates may wake up and try to transmit early ACK packets at the same pause between two preambles. These early ACKs will cause collisions. In order to avoid collisions, each node backoffs a random time before reply an early ACK packet. The longest backoff time must be small so that after the longest backoff, the node can still successfully reply an early ACK packet in the rest time of the pause interval. We also note that the number of candidates affects the E2E delay, the average E2E delay will be reduced if we choose more candidate nodes. We will show how the threshold θ (in fact, α) affects the E2E delay through simulation in section V.

B. Table Initialization and Updating

After nodes have been deployed, sink starts the hop-count value initialization process. In this procedure, sink broadcasts a packet with hop-count value to the sink (sink's hop-count value is 0) and each node rebroadcasts the packet with hop-count value to the sink when receives a broadcasting packet for the first time. Each node only sends out one packet in this procedure, and records the least hop-count value.

After hop-count initialization process finishes, each node starts a neighbor discovery process. The detailed procedure is given as follows:

- Node A broadcasts a neighbor discovery packet with its ID number and hop-count value.
- When neighbor receives this packet, it reads the hop-count from this packet and compares it with its own hop-count. If its hop-count is smaller than or equal to node A's hop-count, it will backoff a random time in order to avoid collisions and then reply an ACK with its hop-count, ID number and RSSI value with the received packet.
- If node receives an ACK packet, it will calculate the LQ (see section III-B) to the response node, and record the ID number, hop-count value and LQ.

Note that only node whose hop-count is no more than node A's hop-count, replies an ACK packet, because we only choose nodes with smaller or equal hop-count as relay. In order to obtain accurate pairwise link quality, each node could broadcasts several neighbor discovery packets. It also solves the problem that some neighbors did not receive the neighbor discovery packet or node A missed some ACKs. When all nodes finish the neighbor discovery process, sink broadcasts a table setting-up packet, it contains its $\widehat{LQ}$ ($\widehat{LQ} = 0$ for sink), ID number and hop-count value. When a node receives a setting-up packet, it calculates the LQ to the sink using Equation 4, and records it in its neighbor table. Once a node has received all the setting-up packets from next-hop neighbors, it broadcasts its setting-up packet with its $\widehat{LQ}$, ID number and hop-count value. Each node will rebroadcast its setting-up packet if its $\widehat{LQ}$ to the sink changes after receiving a setting-up packet from its neighbor. Each setting-up packet can be broadcasted several times in order to make sure that every neighbor can receive at least once the setting-up packets.

During the initialization period, all nodes are in the working state. When neighbor table has been set up, a node starts to operate according to its own schedule. To update the neighbor table, each node contains its $\widehat{LQ}$ and RSSI with the received packet in every ACK if it is chosen as a relay node and receives a packet from its neighbor. Each node receives an ACK can use the $\widehat{LQ}$ and RSSI to update the LQ value in its neighbor table according to the following equation:

$$LQ_i(j) = (1-\beta) * LQ_i(j)_{old} + \beta * (LQ(i,j) + \widehat{LQ_j}) \quad (6)$$

where $LQ_i(j)_{old}$ is the old value of $LQ_i(j)$, and $LQ(i,j) + \widehat{LQ_j}$ is the newly calculated $LQ_i(j)$ value. β is a parameter between $[0,1]$ which affects the convergence rapid of LQ. Here, we let $\beta = 0.8$ in order to quickly reflect network changes.

Note that the updating technique introduced previously only affects good link quality nodes (because only relative good link quality nodes can be chosen as relay nodes). In order to reflect all link quality and network topology changes, each node periodically send out neighbor discovery packet. Once a neighbor with smaller or equal hop-count value receives one such packet, it will reply a packet with its $\widehat{LQ}$, ID number and hop-count value. If node receives these response packets, it will update its neighbor table according to Equation 6.

C. Retransmission and Routing Loops

If we allow a node to endlessly retransmit a packet, it will consume significant energy. Therefore, to set an upper bound of the retransmission time is essential. We denote this upper bound time as Δ. But in our data forwarding scheme, the maximum retransmission time is determined by both Δ and the number of candidate nodes. Equation 7 (given as follows) shows how to determine the maximum retransmission time.

$$\text{maxinum retransimission time} = \min(\Delta, N-1) \quad (7)$$

where N is the number of candidate nodes. For example, we assume that $\Delta = 3$, and node A has 3 candidate nodes, then the maximum retransmission time for node A is twice. We give the transmission process in Fig. 4. Node A has three candidate nodes C1, C2 and C3, when A has data to send, it sends out preambles with the ID numbers of C1, C2 and C3. C1 first wakes up and replies an early ACK, then A sends C1 data packet. If node C1 fails to receive this packet, A will start to retransmit packet to other neighbors (C2 and C3) and C1 will go back to sleep immediately. For A's retransmission, it also begins with preambles. If the retransmission fails to send packet to C2 and C3, A will drop this packet because the maximum retransmission time is twice.

Note that the candidate nodes selection algorithm introduced previously may find out less number of candidate nodes than Δ or even no candidate. Less number of candidate nodes will

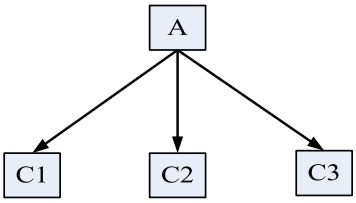


Fig. 4. Data Forwarding Process with Retransmission

reduce the delivery ratio and increase the E2E delay. We solve this problem by choosing the top-Δ neighbor nodes in the neighbor table as candidate nodes. If the number of neighbor nodes in neighbor table is less than Δ, we will choose all the neighbor nodes as candidate.

We note that the forwarding strategy introduced before may cause routing loops, and we also note that routing loops only happen between the same hop-count nodes. In order to detect routing loop, we suppose each packet has a packet number, if a node sends a packet to the same hop-count neighbor, it will record the packet number and the ID number of the original node in its buffer for a period of time. If it receives a packet later has the same packet number and ID number, it will detect that a routing loop happens. When a node detects a routing loop, it will transmit packet only to the first node of its neighbor table, then the first node will transmit packet to the first node of its neighbor table until the packet reaches a next-hop node. Then the next hop node recovers to the DDF forwarding strategy. In short, when routing loop happens, nodes will use ETX routing scheme to forward packet until it reaches a next-hop node, and then recovers to DDF.

V. Simulation and Evaluation

In this section, we provide simulation results to evaluate the performance of DDF. We deploy sensor nodes randomly in a 200m*200m area and the node density is $8\ nodes\ per\ 1000m^2$. A sink node is located at the center of the deployment area and each node randomly generates packet and sends it to the sink over multiple hops. We also set the threshold $\alpha = 0.6$, the duty cycle equals 5% and the initial energy of each node equals 1 and the energy consumption of packet transmission and reception equals 0.00015 and 0.0001, respectively. In order to get more realistic performance, we use the link model introduced in section III-B to determine link quality. We also use the method proposed in [12] to calculate SNR and set the parameters of the link model with respect to Chipcon CC1000 [15]. We compare the performance of DDF with ETX. Unless otherwise specified, each experiment is repeated 40 times with random node deployment under these parameters.

Fig. 5 shows the performance under different duty cycles. Fig. 5(a) gives the average delivery ratio, we can see that both DDF and ETX have steady delivery ratio under fixed retransmission time. We denote the retransmission time $\Delta = 0$, $\Delta = 1$, $\Delta = 2$ for ETX and DDF as ETX0, DDF0, ETX1, DDF1, ETX2 and DDF2, respectively. The average delivery ratio for ETX0, ETX1, ETX2, DDF0, DDF1, DDF2 is 86.37%, 97.5%, 99.33%, 83.2%, 99.1% and 99.76%, respectively. DDF achieves a little higher delivery ratio than ETX when retransmission is allowed. This is because those packets who meet routing loops have more chance to be retransmitted. We observe that there are about 2-6% packets meet routing loops. However, when we do not allow retransmission, the average delivery ratio for ETX is higher than DDF. This is because ETX always forwards data along the best routing path while DDF forwards data along relative better routing path. Fig. 5(b) gives the E2E delay of ETX and DDF, DDF can reduce E2E delay about 38-55%. Fig. 5(c) shows the lifetime improvement for DDF compared with ETX (the lifetime of ETX is denoted as 1 for all duty cycle). We can see that the lifetime improvement increases with the duty cycle increases, from about 11% to 36%, because more nodes will wake up at a short period of time with the increase of duty cycle, this can balance energy consumption among these nodes.

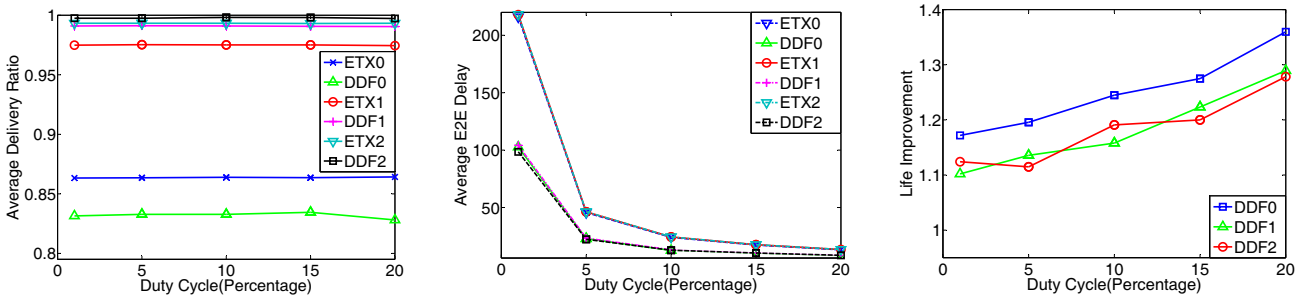
We only provide the performance of DDF under different threshold value of α in Fig. 6, since α has no impact on ETX. The larger threshold α is, the higher link quality of candidate nodes is and the smaller number of candidate nodes is. So the delivery ratio (see Fig. 6(a)) and E2E delay (see Fig. 6(b)) increases when α increases. But the E2E delay under $\alpha = 0.5$ is higher than $\alpha = 0.6$, because more retransmissions are needed when the link quality of candidate nodes is low. The lifetime improvement of DDF comparing with ETX is given in Fig. 6(c). Lifetime improvement increases with the increase of α when retransmission is allowed, because the larger α is, the higher link quality is. Choosing high link quality nodes as relay nodes can reduce the retransmission time, so more energy is saved. But if we forbid retransmission, the lifetime improvement will decrease, because the energy consumption burden for those good link quality candidate nodes increases.

Fig. 7 presents the performance under different node densities. The larger node density is, the more good link quality candidate nodes will be found. So the delivery ratio (see Fig. 7(a)) increases and the E2E delay (see Fig. 7(b)) decreases when node density increases. In order to compare lifetime improvement, we denote the lifetime for ETX under node density of $6\ nodes\ per\ 1000m^2$ as 1. From Fig. 7(c), we can see that with the increase of node density, the lifetime improvement for ETX0, ETX1 and ETX2 increases from 0 to 35.2%, 37.6% and 40%, respectively, while the lifetime improvement for DDF0, DDF1 and DDF2 increases from 24.7%, 14% and 11.7% to 72%, 64.8% and 64.7%, respectively.

The results of our simulation demonstrate that our solution DDF outperforms ETX. It delivers almost 100% of packets when retransmission is allowed, reduces the E2E delay by about 50% and increases the network lifetime by about 10-35%.

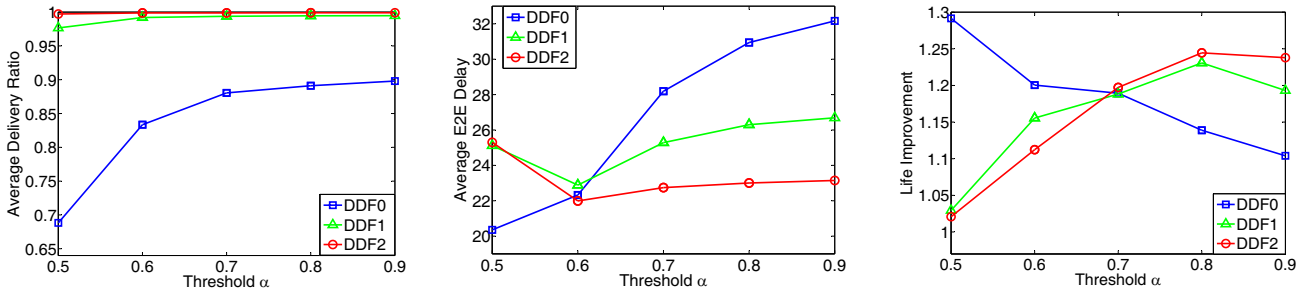
VI. Conclusion

In this paper, we propose a dynamic data forwarding (DDF) scheme for low-duty-cycle WSNs, which combines a realistic



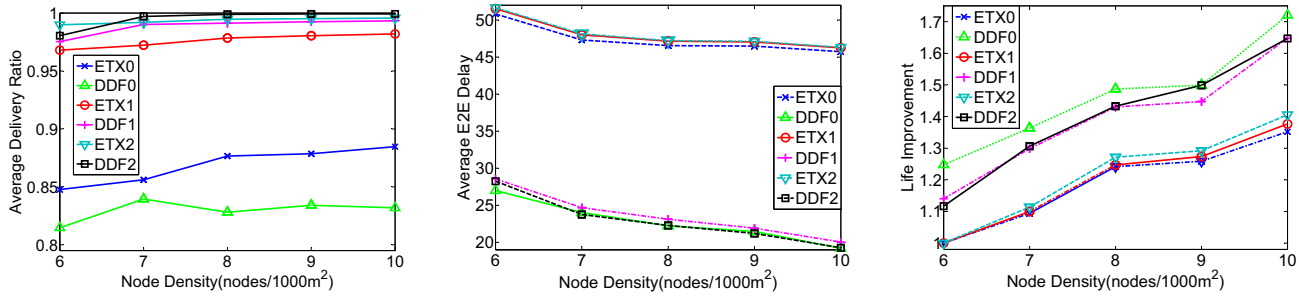
(a) Delivery Ratio vs. Duty Cycle (b) E2E Delay vs. Duty Cycle (c) Lifetime vs. Duty Cycle

Fig. 5. Performance under Different Duty Cycle



(a) Delivery Ratio vs. Threshold (b) E2E Delay vs. Threshold (c) Lifetime vs. Threshold

Fig. 6. Performance under Different Threshold



(a) Delivery Ratio vs. Node Density (b) E2E Delay vs. Node Density (c) Lifetime vs. Node Density

Fig. 7. Performance under Different Node Density

link model with asynchronous duty cycle. To evaluate the performance of DDF, we have performed extensive simulations. The results show that DDF can reduce E2E delay, guarantee delivery ratio and improve network lifetime compared with ETX. In the future, we intend to extend this work into large scale WSNs with multiple sinks and even with mobile sinks. We also intend to implement DDF on real testbeds and evaluate the performance.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality of the paper. This work is partly supported by China NSF grants (60825205, 60903179, 61073152, 61170236, 61133006).

REFERENCES

[1] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *In Proceedings of the Second International Conference On Embedded Networked Sensor Systems (SenSys 2004)*, p95-107, November 2004.

[2] W. Ye, John S. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, p1567-1576, June 2002.

[3] T. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *In Proceedings of the First International Conference On Embedded Networked Sensor Systems (SenSys 2003)*, p171-180, November 2003.

[4] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks," *In Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, p307-320, 2006.

[5] B. Karp and H.T. Kung, "Gpsr: Greedy Perimeter Stateless Routing for Wireless Networks," *In Proceedings of the 6th Annual MOBICOM, ACM Press*, p243-254, 2000.

[6] R. Fonseca, S. Ratnasamy, J. Zhao, C.T. Ee, D. Culler, S. Shenker and I. Stoica, "Beacon Vector Routing: Scalable Point to Point Routing in Wireless Sensornets," *In Proceedings of the Second USENIX/ACM Syposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[7] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-demand Distance Vector Routing," 2003, RFC 3561.

[8] D.B. Johnson, D.A. Malz, and J. Broch, "DSR: the Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks," *Ad hoc networking*. p139-172, 2001.

[9] J. Kim, X. Lin, and N.B. Shroff, "Optimal Anycast Technique for Delay-Sensitive Energy-Constrained Asynchronous Wireless Sensor Networks," *In Proceedings of the 28th Conference on Computer Communications (INFOCOM)*, p612-620, 2009.

[10] R.R. Choudhury and N.H. Vaidya, "MAC-Layer Anycasting in Ad Hoc Networks," *SIGCOMM Computer Communication Review*, vol. 34, p75-80, January 2004.

[11] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," *In Proceedings of the Second International Conference on Mobile Systems, Applications and Services (MOBYSIS)*, 2004.

[12] M. Zuniga, and B. Krishnamachari, "Analyzing the Transitional Region in Low Power Wireless Links," *In Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004*, p517-526, 2004.

[13] D.S.J.D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High Throughput Path Metric for Multi-Hop Wireless Routing," *In Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, p134-146, 2003.

[14] Y. Gu, and T. He, "Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links," *In Proceedings of the First International Conference On Embedded Networked Sensor Systems(SenSys)*, 2007.

[15] Chipcon. CC1000 low power radio transceiver, http://focus.ti.com/.

[16] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, "Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks," *In Technical Report UCLA/CSD-TR* p02-0013, 2002.