Delay-aware Routing in Low Duty-Cycle Wireless Sensor Networks

Guodong Sun Bin Xu
Computer Science and Technology Department
Tsinghua University, Beijing, China, 100084
gdsun@ieee.org
xubin@tsinghua.edu.cn

***Abstract*—In low duty-cycle sensor networks, sensors sleep most of their time; it is consequently hard for sensors to maintain always-awake communication links. One major challenge is therefore to design communication algorithms to reduce the delivery delay. In this paper we propose and evaluate a delay-aware routing algorithm for low duty-cycle sensor networks. The proposed algorithm first organizes the network into a layered structure. And then the sensor with packets ready to be sent dynamically determines a forwarder from multiple potential next-hop sensors in terms of timeliness. Simulation results show that the proposed algorithm outperforms the static shortest-path routing in delivery delay and drop rate.**

***Index Terms*—Sensor networks, low duty cycle, routing, delay.**

I. Introduction

Advances in microelectronics, wireless networking and embedded processor make wireless sensor networks applicable to both civilian and military applications [1], [2], [3]. Sensor nodes work relying on capacity-small, unrechargeable batteries, and communicate with each other via limited bandwidth. To save sensors' energy and then prolong the system lifetime, it is efficient for sensors to work in a low duty-cycle fashion where they sleep most of their time and wake up to receive data only in a low percentage of their time. Low duty-cycle sensor networks effectively reduce the energy consumed in idle listening, packet overhearing and control overhead. However, one problem experienced by low duty-cycle sensor networks is the long delivery delay caused by the sleep latency of sensors. In many sensor network scenarios, including military surveillance, target tracking and monitoring, the delay is critical to the performance of systems.

In this paper we focus on designing a delay-aware routing algorithm for low duty-cycle sensor networks to reduce the network delay. The proposed algorithm is simple but effective to improve the delay performance while reducing the data packet drop rate over networks.

The rest of the paper is organized as follows. Section II introduces some concepts and the network model used in this paper. The details of our algorithm can be seen in section III. Section IV compares our algorithm and a baseline algorithm by extensive simulations. Section V briefly summarizes the related work. Finally, we conclude our work in section VI.

This work is supported in part by the National Science Foundation of China under Grant No.60803124.

II. Network Model

We assume that the sensor network consists of n sensor nodes randomly deployed in a $L \times L$ square. Every sensor runs on the radio range r. In real cases, the communication range of sensor nodes is far from being a circle with some fixed radius. The radio range varies due to many factors, such as channel interferences, noises, directional antennas, and so on. Therefore two sensors can communicate if and only if the distance between them is equal to or less than r and the bidirectional link quality between them is beyond some threshold. We assume that $r < L$ and the network is always of multi-hop and that the network is connected.

A. Duty Cycle of Sensors

For all duty-cycle sensors, there are two states: *active* or *awake*, and *sleeping*. In active state, the sensor can sense and receive data from its neighbors. When a sensor is in the sleeping state, it turns off all its function modules except a timer to wake itself up, or turns off only its transceiver module while being able to process data. The working time of a sensor is often divided into periodic working schedules. For instance, in Fig.1, each working schedule of sensors A and B involves five slots(time units). The duty cycle is defined as the ratio between the active time and the total time of a working schedule. In Fig.1, the duty cycles of A and B are both 20%. In low duty-cycle sensor networks, the sleeping time is always much longer than the active time.

As illustrated in Fig. 1, a single working schedule can be represented by a finite binary(0-1) string in which 1 represents the active state and 0 the sleeping state. A sensor can exchange its working schedule of format binary string with its neighbors; furthermore it can broadcast only the active bits, e.g. the message containing $(100|1, 45)$ which means the working schedule of A is of length 100 slots and the 1st and 45th slots are active states, consequently reducing the communication overhead among sensors.

B. Channel Access

In many wireless systems, the same wireless channel or sub-channel is shared by multiple peers. Therefore, some proper channel access control is required to handle the contention. In the paper, a simplified *Protocol Model* [4] is used to control channel access. Consider a transmission from node A to node

978-1-4244-3709-2/10/$25.00 ©2010 IEEE

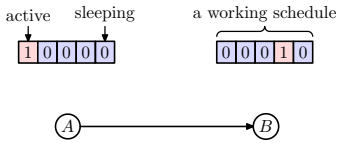


Fig. 1. Working schedule

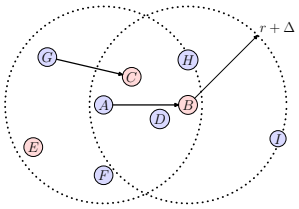


Fig. 2. Channel interference

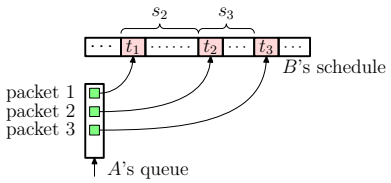


Fig. 3. Queuing delay

B. Let X_A and X_B represent the locations of A and B, respectively. This transmission can be successfully received by B if the following two conditions hold:

1) the distance between A and B is no more than r, i.e. $|X_A - X_B| \leq r$, and
2) for every other node X_K who is simultaneously transmitting over the same channel, $|X_K - X_B| > r + \Delta$ where Δ weighs the radio interference.

Obviously, both transmissions of A to B and G to C in Fig. 2 can not be carried out simultaneously. We assume that a CSMA/CA like method is used. When A wants to send data to B, A firstly listens the channel. If the channel is busy, A differs its data transmission and sleeps until B's next active slot; otherwise it sends a request message `REQ` to B. If A receives the reply message `CLR`, it will start its transceiver to send data.

C. Delay Model

To model the per-hop delay in any wireless networks, three delay components are involved: (1) queuing delay: the waiting time of one packet in buffer to be scheduled; (2) transmission delay: the time taken by operating systems to move one packet from the buffer to the transceiver; and (3) propagation delay: the time taken by data propagated between two transceivers.

For low duty-cycle sensor networks, one packet in buffer queue has to wait until one of its next-hop sensors enters the active state. Consider a transmission from sensor A to sensor B shown in Fig. 3, where packet 1 queued at transmitter A is forwarded at time t_1 when the receiver B is active. Because only one packet can be transmitted at any slot, packet 2 has to wait until the next active state comes to B. As mentioned previously, sensors in low duty-cycle networks spend most time in sleep; therefore, packet 2 can not yet be transmitted at slot $t_1 + 1$ even after the packet 1 has arrived at B within slot t_1; packet 2 has to wait until slot t_2 comes. Similarly, packet 3 can only use slot t_3, instead of slot $t_2 + 1$. Obviously, the transmission delay and propagation delay, compared with the queuing delay, therefore can both be neglected. As a result we consider only the queuing delay reduction in our algorithm design. In this paper, let τ denote the time span of a time slot.

III. Algorithm Design

Our algorithm involves two phases: network initializing and dynamic forwarding. In the first phase, the random network is organized into a layered structure where every sensor is assigned a layer number. In the second one, the sensor with layer i dynamically selects its next hop in terms of timeliness from its neighboring sensors with layer $i - 1$. .

A. Network Initializing

We first organize the network topology into a layered structure. The left figure in Fig.4 shows five sensors and the links among them, and the right shows the corresponding layered structure where every sensor can only forward data to next-layer sensors,e.g. B can forward data to S, but not to C or D. We will next briefly introduce the procedure of constructing the layered topology.

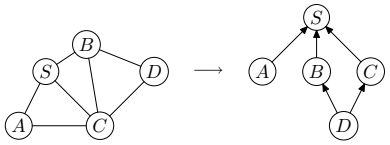


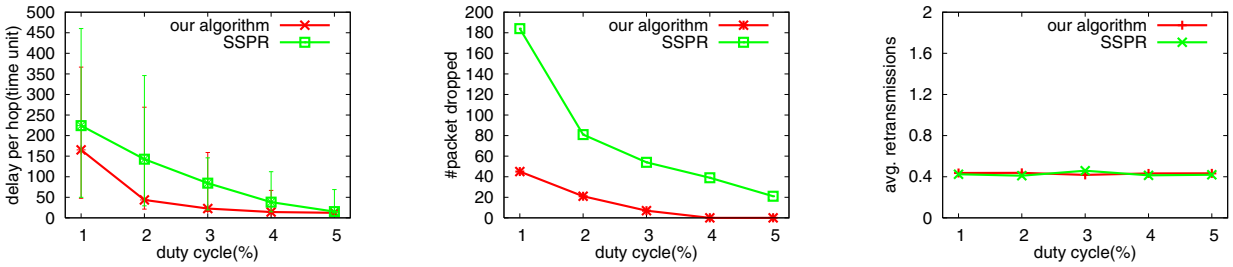
Fig. 4. Layered topology

After being deployed, sensors can determine their neighbors by broadcasting and receiving messages containing ID and working schedule information. With the neighborhood knowledge, a flooding method can be used to construct layered topology. Initially, the sink node broadcasts a message containing its layer number, 0. If sensor B receives a message from S, B sets its layer with 1. Similarly, A as well as C is of layer 1. If B receives a message from A or C, it does not update its layer number. In our algorithm, the layer-i sensor can only forward data to its neighbors in the $(i-1)$-th layer. We allow multiple potential forwarders at each hop, for instance, D can forward data to B or C. In flooding process, a challenge is to avoid the flooding storm, and how to design and implement an effective flooding for low duty-cycle sensor networks can be seen in [5].

B. Dynamic Forwarding

After being layered, every sensor can generate a forwarding set if the network is connected. The forwarding set of sensor A, denoted as F_A includes all the sensors that can relay data for A. In low duty-cycle sensor networks, a sensor sleeps most time and consequently it can not always forward data whenever its previous-hop sensor wants to send data. In order to reduce the delay caused by the sleeping latency, one sensor to send data should prefer forwarders that are upcoming to wake up. We take a walk-through of our dynamic forwarding policy as follows.

In Fig. 4, $F_D = \{B, C\}$. If D knows the working schedules of B and C, it can create a forwarding sequence set, denoted as S_D by ascendingly sorting the nodes in F_D in the order of the wake-up time. Suppose that upon time t_0, D has three



(a) Delay per hop (b) Avg. drops per running (c) Avg. retransmissions per packet

Fig. 5. Performance under different duty cycles. All simulations run on a 150 × 150 square with 150 sensors.

packets ready to be sent and that the working schedules of B and C are $(100|5, 30, 62)$ and $(100|3, 24, 30)$, respectively. Then we have $S_D = \{C, B, C, C, B, B\}$ shown in Fig.6.

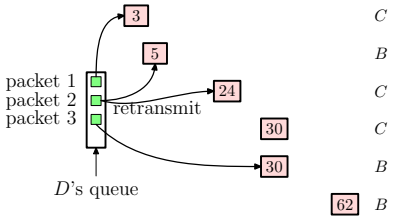


Fig. 6. The procedure of dynamic forwarding

Fig.6 shows the dynamic forwarding process. D sends packet 1 at time t_1 when C's 3rd slot is active; similarly packet 2 is sent at time t_2 when B's 5th slot is active. Note that packets are often lost or damaged due to the lossy, noisy link. Once a transmission is failed, the corresponding packet should be retransmitted. In Fig.6, the transmission of packet 2 toward B fails and the following retransmission is directed to C, no longer to B because C wakes up earlier than B. If D can not send all its three packets in the duration of a working schedule, it just wait the next working schedule without adjusting its forwarding sequence set. When two forwarders enter active states simultaneously, the transmitter selects only one as forwarder. In Fig. 6, the 30th slots both of B and C are active, D forwards packet 3 to B that has more energy, more buffer space, or higher link quality. In our implementation, among the forwarders with identical awake slot the forwarder with the highest bidirectional link quality is preferred.

IV. Simulation Evaluation

We perform simulations to evaluate our algorithm. In the simulator, the radio layer is implemented according to [6] in which the dynamic property of radio links is considered. Additionally, we validate the links with the link quality greater than or equal to 0.5. All results are averages of 100 runs. We use the static shortest-path routing(SSPR) as the baseline algorithm in which every sensor selects randomly a forwarder during initialization and uses this forwarder all the time.

A. Simulation setup

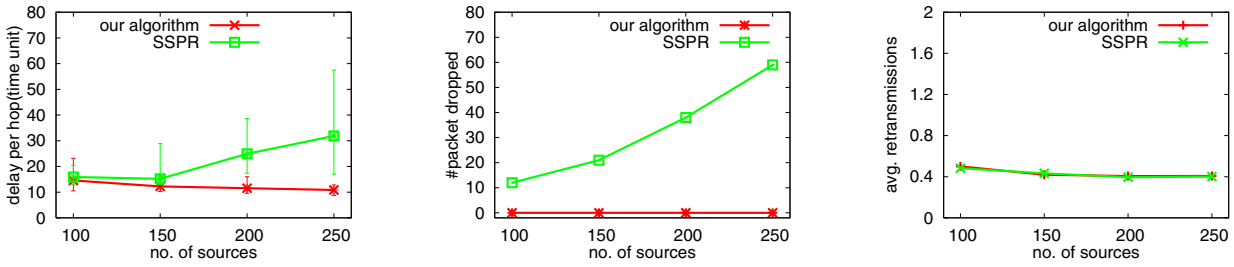
Let L be 150 in fault and the sink node centers the square. The working schedule of each sensor is length 100τ and $1\tau = 20$ms. In every slot, the sensor generates one packet with probability 2×10^{-4}, that is, a sensor can generate three packets every five minutes in average. Every simulation runs $10^5\tau$. In all simulations, the parameter of maximum allowed retransmissions in MAC layer is set to be four beyond which the relevant packet will be dropped.

B. Experimental Results

We first compare the per-hop delay between our algorithm and SSPR under different duty cycles; the related results are plotted in Fig.5. We can see that the average delay in both algorithms decreases with the duty cycle increasing, and the average delay of SSPR is obviously greater than ours until the duty cycles increases to 5%. Note that the maximum and minimum delay occurred in experiments of SSPR is significantly greater than the two counterparts of ours; in other words, our algorithm is more stable than SSPR in delay performance. Since our algorithm always send data to the awake-earliest potential forwarder, the packets in buffer are queued up slowly while in SSPR, the packet of a sensor has to wait until its forwarder enters active state; therefore our algorithm achieves a lower drop rate over the network. Since we use a CSMA/CA-like MAC and particularly, the packet generation rate is very low due to the capacity limitation of low duty-cycle sensor networks, the retransmission rate in both algorithms is identical and stays about 0.4, i.e. every packet received by the sink has been retransmitted 0.4 times.

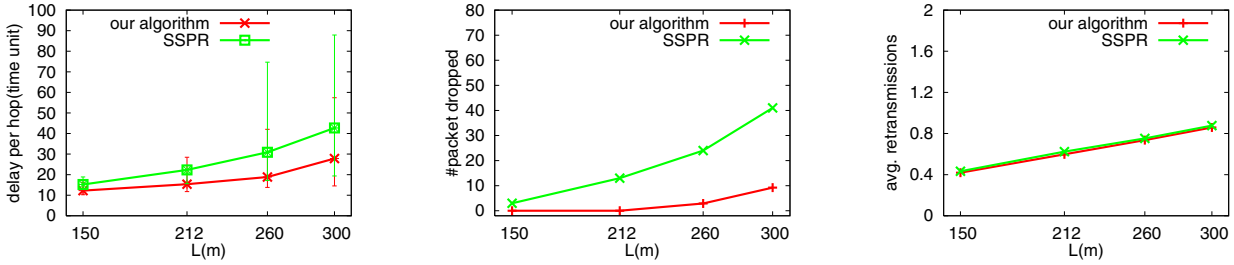
The three figures in Fig.7 plot the performance comparison under different sensor densities. As shown in Fig.7a, the per-hop delay in both algorithms decrease a bit when the number of sensor increases from 100 to 150. The reason behind is that more sensors improve the connectivity and consequently each sensor has more potential forwarders such that the sensor just has to wait less time. But the delay of SSPR dramatically increases when the sensor quantity increases from 150 to 250 while our algorithm remains at a per-hop delay of 15τ. Fig.7b shows the drop rate under SSPR climbs up while ours still remains zero. Two algorithms achieves identical retransmissions illustrated in Fig.7c

Finally, we investigate the performance of both algorithms under different network scales, but identical sensor density. Fig.8 gives the related results. If the network area changes larger, the number of sensors should be increased in order to maintain a constant density. Consequently the delay and



(a) Delay per hop (b) Avg. drops per running (c) Avg. retransmissions per packet

Fig. 7. Performance under different sensor densities. All simulations run on a 150×150 square with different number of sensors.



(a) Delay per hop (b) Avg. drops per running (c) Avg. retransmissions per packet

Fig. 8. Performance under variable network scale. In all simulations, the sensor density is always 8 and the duty cycle is 5%.

the drop rate in both algorithms rise with the network scale increasing because more sensors with limited buffer space will share the same channel. Of course the retransmission rate also increases slightly in both algorithms, but their retransmission rates are still identical.

V. Related Work

In the past few years, the low duty-cycle sensor network has gained more attention. We summarize some significant related work as follows.

SCP [7] is a MAC scheme designed for ultra-low duty-cycle(0.1%) sensor networks. SCP employs LPL(low-power listening) technique and extends the system lifetime of a sensor network by a factor of 3 to 6 times. DSF [8] considers both the lossy channel and delay resulted from low duty-cycle sensor networks. DSF includes three optimizations: delivery ratio, end-to-end delay, and energy consumption. Differently than DSF, the forwarding sequence defined in this paper allows repeated forwarders only if their wake-up times are different. Guo Shuo et al. [5] proposed an opportunistic flooding algorithm which reduces the transmission redundancy and achieves fast dissemination. Similarly, ADB [9], another flooding algorithm for asynchronous duty-cycle sensor networks was designed. In [10], three approaches were proposed to improve the real-time capacity of low duty-cycle sensor networks by adding more active states in nodes' working schedules. The work in [11] proposes QoS definition in low duty cycle sensor networks and analyze the effect of duty cycling for reaching the availability and reliability.

VI. Conclusion

In this paper, we propose a delay-aware routing algorithm for low duty-cycle sensor networks to address the sleep latency in data forwarding. Differently than traditional static routing algorithms, our algorithm achieves shorter delay by dynamically selecting forwarders. In our algorithm, every sensor maintains a forwarding sequence set in which potential forwarders are sorted in the order of their wake-up time. The simulation results demonstrate that our algorithm improves the delivery delay over the network, and reduces the network drop rate, correspondingly saving the energy of sensors. In future work, we will evaluate our algorithm under scenarios with time-varying links.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–442, 2002.

[2] D. Culler, D. Estrin, and M. Srivastava, "Overivew of sensor networks," *IEEE Computer Magazine*, 2004.

[3] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, pp. 2292–2330, 2008.

[4] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Technology*, vol. 46, no. 2, pp. 388–401, March 2000.

[5] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links," in *SenSys*, 2009, pp. 133–144.

[6] M. Zuniga and B. Krishnamachari, "Analyzing the transmissional region in low power wireless links," in *IEEE SECON*, 2004, pp. 517–526.

[7] W. Ye, F. Silva, and J. Heidemann, "Ultra-low dudty cycle mac with scheduled channel polling," in *SenSys*, 2006.

[8] Y. Gu and T. He, "Data forwarding in extremely low-duty-cycle sensor networks with unreliable communication links," in *SenSys*, Sydney, Australia, 2007, pp. 321–334.

[9] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. B. Johnson, "Adb: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks," in *SenSys*, 2009, pp. 43–56.

[10] Y. Gu, T. He, M. Lin, and J. Xu, "Spatiotemporal delay control for low-duty-cycle sensor networks," in *RTSS*, 2009.

[11] J. Suhonen, T. D. Hamalainen, and M. Hannikainen, "Availability and end-to-end reliablity in low duty cycle multihop wireless sensor networks," *Sensors*, vol. 9, pp. 2088–2116, 2009.