LETTER

A Distributed TDMA-Based Data Gathering Scheme for Wireless Sensor Networks*

Tao LIU[†,††a)], *Member*, Tianrui LI[†], *and* Yihong CHEN[†,††], *Nonmembers*

SUMMARY In this letter, a distributed TDMA-based data gathering scheme for wireless sensor networks, called DTDGS, is proposed in order to avoid transmission collisions, achieve high levels of power conservation and improve network lifetime. Our study is based on corona-based network division and a distributed TDMA-based scheduling mechanism. Different from a centralized algorithm, DTDGS does not need a centralized gateway to assign the transmission time slots and compute the route for each node. In DTDGS, each node selects its transmission slots and next-hop forwarding node according to the information gathered from neighbor nodes. It aims at avoiding transmission collisions and balancing energy consumption among nodes in the same corona. Compared with previous data gathering schemes, DTDGS is highly scalable and energy efficient. Simulation results show high the energy efficiency of DTDGS.

key words: *wireless sensor networks, data gathering scheme, distributed TDMA scheduling*

1. Introduction

In a large-scale periodic data-gathering wireless senor network (WSN), hundreds or even thousands of sensor nodes may be dispersed over the monitoring area, and each node must periodically report its sensed data to the sink through single-hop or multihop wireless routing. Its applications include remote habitat monitoring, battlefield monitoring, environmental data collection, etc. However, frequent information collection in periodic data-gathering sensor networks leads to a tremendous traffic burden on the networks and serious transmission collisions. Due to its collision-free and energy efficient properties, time division multiple access (TDMA) has been widely adopted in WSNs [1]. An earlier approach [2] proposed a TDMA-based scheduling scheme that balances the energy-saving and end-to-end delay in a centralized approach. In [2], the gateway gathers the connectivity information from sensor nodes to construct a TDMA frame that ensures collision avoidance. Finally, this schedule is flooded back to the nodes. S-MAC [3] negotiates a schedule that specifies when nodes are awake and asleep within a frame. Reference [4] proposes an address-light, integrated MAC and routing protocol (AIMRP). AIMRP is an integrated MAC and routing mechanism designed specifically for WSNs which have to promptly detect and report relatively rare events. These existing algorithms are either centralized with poor scalability or unsuitable for the periodic data-gathering WSNs.

In this letter, a distributed TDMA-based data gathering scheme (DTDGS) is proposed in order to avoid transmission collisions, achieve high levels of power conservation and improve network lifetime in periodic data-gathering WSNs. In DTDGS, each node builds its own schedule and selects its next-hop forwarding node according to the information gathered from neighbor nodes. The DTDGS does not need a centralized gateway to assign transmission slots and the next-hop forwarding node to every node. DTDGS is inherently collision-free, highly scalable and energy efficient, thus apt for large-scale periodic data-gathering WSNs.

2. Network Model

Similar to the model in [4], we assume that all sensor nodes are uniformly distributed in a circular monitoring area A of radius L with node distribution density ρ. There is only one sink which is located at the center of A. All sensor nodes have the same transmission range r. The entire network is organized into K concentric coronas centered at the sink with the same width r. The coronas are numbered $C_1, C_2, \ldots, C_K$ starting from the innermost corona, as shown in Fig. 1. Nodes in the corona C_i should forward their data to the nodes in corona C_{i-1} $(1 < i \leq K)$, and the nodes in C_1 communicate directly with the sink. So a node in C_i can relay a message to the sink in i hops.

We assume every node has a unique node ID and knows

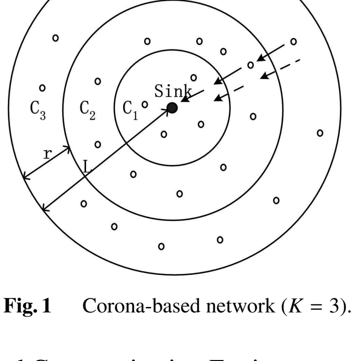

Fig. 1 Corona-based network ($K = 3$).

Manuscript received February 18, 2013.
Manuscript revised May 12, 2013.

[†]The authors are with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China.
[††]The authors are also with the School of Computer Science and Technology, Southwest University for Nationalities, Chengdu, China.

[*]This research was Supported by the Fundamental Research Funds for the Central Universities, Southwest University for Nationalities (13NZYTD02).

a) E-mail: liucomputer@189.cn

DOI: 10.1587/transinf.E96.D.2135

the node ID of its neighbor nodes in the next inner corona. Besides, all nodes in the network can be synchronized.

3. The DTDGS Scheme

The operation of DTDGS is divided into two phases: schedule set-up phase and data dissemination phase. In the schedule set-up phase, the transmission schedule is built. After the construction of the schedule, the network enters the data dissemination phase. The nodes send and receive data according to the schedule in this phase. Detailed descriptions of these two phases are in the following two subsections.

3.1 Data Dissemination Phase

In data dissemination phase, all nodes periodically report data in every data gathering period. To avoid collisions, we introduce a two-level TDMA-based scheduling mechanism into the DTDGS scheme.

In Level-1 scheduling, we divide a data gathering period into K sub-periods. Each corona is assigned to a sub-period. As shown in Fig. 2, the nodes in corona C_i ($1 < i \leq K$) should transmit their sensing and relayed data in the $(K-i+1)th$ subperiod while the nodes in C_{i-1} receive these data. The nodes in C_1 should transmit their data to the sink directly in the Kth subperiod. To save energy, the nodes should go into sleep mode when they have no data to send or receive.

In Level-2 scheduling, each sub-period is further divided into m time slots, as indicated in Fig. 2. m is large enough to ensure that each node can send its data and the length of a slot is long enough to send a data packet. In a data gathering period, each node sends data in its transmission slots of corresponding sub-period. Note that the transmission slots of a node are assigned in the schedule set-up phase. Obviously, all the sensor nodes are in the sleep mode most of the time in a data gathering period.

3.2 Schedule Set-Up Phase

In schedule set-up phase, each node chooses its transmission slots and next-hop forwarding node according to the collected information gathered from neighbor nodes in the next inner corona. Different from the centralized approach, the transmission schedule is built by a distributed algorithm. Obviously, the DTDGS scheme has better scalability than the centralized approach.

To build the transmission schedule that ensures collision avoidance and choose the next-hop forwarding node that achieves energy balance, each node should create a table T, which depicts the TDMA schedule of its neighbor nodes in the next inner corona. An example of a node's table T is illustrated in Table 1. R indicates the index number corresponding to a specific time slot that the neighbor node will switch on its receiver. Note that the total number of R of a neighbor node is equal to the load of this node. The information contained in table T is required for building the

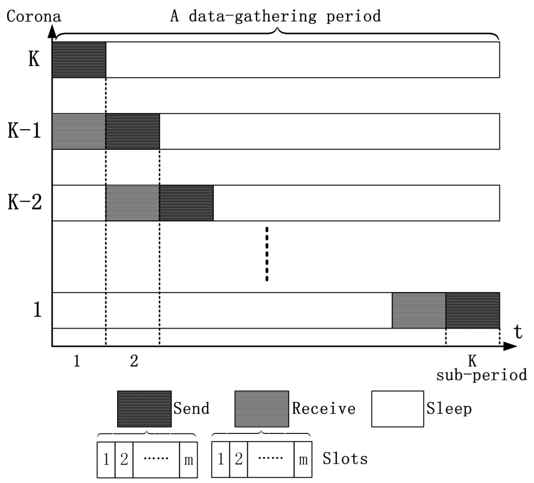

Fig. 2 Temporal division in a data gathering period.

Table 1 An example of a node's table T.

NodeID \ slot	1	2	3	4	5	6	7……m
3	R	R					
4			R		R		
7	R						
8		R		R		R	
……….							

transmission schedule and selecting the next-hop destination node. The following subsections show how table T is created and the transmission schedule is built.

3.2.1 RTS/CTS Handshaking

Initially each node creates its table T and add the information of its neighbor nodes in the next inner corona into table T. The schedule set-up phase is divided into K subphases. To avoid collisions, each node in C_i ($1 \leq i \leq K$) takes it in turn to choose the transmission slots and next-hop forwarding node in the $(K-i+1)th$ subphase. The building of transmission schedule is based on RTS/CTS handshaking.

We demonstrate the operation of the proposed RTS/CTS handshaking protocol with an example. As shown in Fig. 3, node A is located in corona C_i and begin to choose the transmission slots and next-hop forwarding node. Considering the energy balance among nodes in a corona, node A selects its next-hop forwarding node B from its neighbor nodes in table T which has the minimum load firstly. Then, A chooses transmission slots from its table T. To avoid transmission, it is ensured that no one neighbor node of the next inner corona receives data in the transmission slots of node A. Since a data packet is sent in one transmission slot, the number of slots required by node A is equal to the number of data packets transmitted by node A, including sensing and relayed data. Secondly, A broadcasts a RTS packet which contains the index numbers of its transmission slots and next-hop forwarding node ID. As shown in Fig. 3, after the next-hop forwarding node B in C_{i-1} receives this RTS packet, it recomputes its load and replies with a CTS packet.

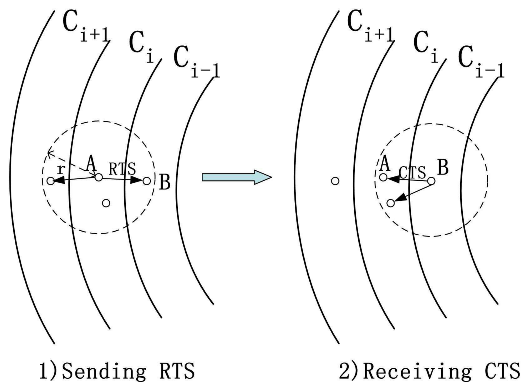

Fig. 3 RTS/CTS handshaking protocol.

Algorithm 1: Selecting transmission time slots and next-hop node

```
On its slice
    At the beginning of slice
        Select next-hop forwarding node B which has the minimum-
            load from its table T;
        Select transmission time slots (ReqSlot1,ReqSlot2,...,ReqSlotl)-
            that ensures collision avoidance;
        NextHopNodeID=B.NodeID;
        Send RTS(NextHopNodeID,ReqSlot1,ReqSlot2,...,ReqSlotl);
    On receiving CTS from Node B
        if B.ReplytoNodeID==NodeID
            SendSlot={ReqSlot1,ReqSlot2,...,ReqSlotl};
            Go into sleep mode;
            Exit;
        else
            T(B.NodeID).slot[B.ReplySlot1]=R; //Update table T
            ......
            T(B.NodeID).slot[B.ReplySlotl]=R;
        end if
End
```

Algorithm 2: Receiving RTS and sending CTS

```
On receiving RTS from node A
    if A.NextHopNodeID= NodeID
        ReplytoNodeID = A.NodeID;
        ReplySlot1 = A.ReqSlot1;
        ......
        ReplySlotl = A.ReqSlotl;
        load = load + l;
        Send CTS(ReplytoNodeID, ReplySlot1, ..., ReplySlotl);
    end if
```

A receives this CTS packet and go into sleep mode until the next subphase. Other nodes in C_i receiving this CTS packet update its table T. Note that the nodes in C_1 send RTS to the sink and sink replies with a CTS packet.

After each node in C_i builds its transmission schedule according to the information contained in its table T, each node in C_{i-1} will know the number of its relayed data packets. Repetition goes on from the outmost corona to the innermost one until all nodes in the network have built their transmission schedules and chosen their next-hop forwarding nodes.

3.2.2 The Distributed TDMA Scheduling Algorithm

In the ith subphase of schedule set-up phase, the subphase is divided into several slices. A slice is assigned to every node in corona C_i. In the slice assigned to a node A in C_i, the pseudocode for node A is given in Algorithm 1. Algorithm 1 consists of two parts. In the first part, at the beginning of slice, node A selects the next-hop forwarding node and transmission time slots according to the information contained in its table T. l denotes the number of slots required by node A and can be computed with the load of node A ($l = load + 1$). Then, node A broadcasts a RTS packet which contains index numbers of its transmission slots and next-hop forwarding node ID. In the second part, when A receives the CTS packet, it decides whether this CTS packet replies to itself. If this packet replies to A, it means A can send data in the time slots required by itself. Then, A goes into sleep mode until the next subphase. *ReqSlotx* and *SendSlot* denote the required slot index number and sending slot index number, respectively. However, if this CTS packet replies to other node, its table T is updated according to the information contained in this CTS packet.

When the nodes in corona C_i build their transmission schedules in the $(K - i + 1)th$ subphase, the nodes in C_{i-1} have to keep listening the full time of this subphase. The pseudocode for the node in C_{i-1} is given in Algorithm 2.

In Algorithm 2, if a node in C_{i-1} receives a RTS packet from node A, it will judge from the information contained in RTS packet. If it is the next-hop forwarding node of node A, it sends a CTS packet to inform A and other neighbor nodes in C_i.

4. Computation of *m*

The nodes in the inner coronas not only transmit their own sensing data, but also relay the data from other nodes in the outer coronas. Therefore, a node in a inner corona usually has to relay more data compared with a node in a outer corona. Obviously, several nodes in a inner corona need more slot for transmitting their data, especially the nodes in the first corona. Besides, the sink can not receive the data from different nodes in the first corona simultaneously. Therefore, the nodes in the first corona need the most slots to transmit their data packets, including their sensing data and relayed data. Since each node generates a data packet in every data gathering period, the total number of data packets transmitted by the nodes in C_1 in a data gathering period is $\rho\pi L^2$. Therefore, each node can achieve enough slots to transmit its data only when

$$m \geq \rho\pi L^2 \tag{1}$$

5. Performance Results

To validate the performance of the DTDGS scheme, we simulate a homogeneous WSN consisting of N sensor nodes uniformly deployed in a circle region. Unless otherwise

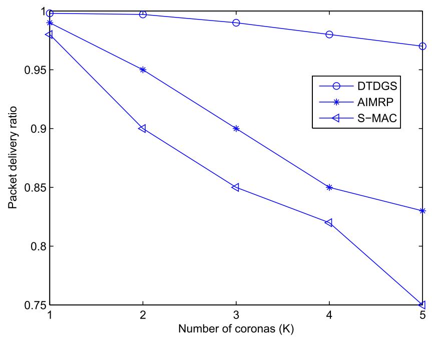

Fig. 4 Packet delivery ratio vs. number of coronas (K).

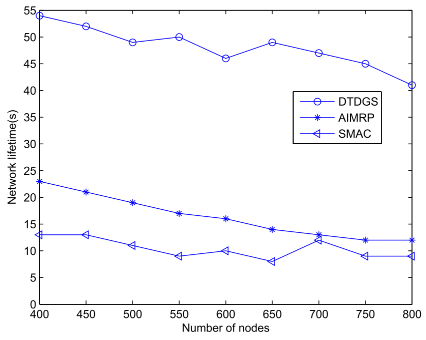

Fig. 5 Network lifetime vs. number of nodes (N).

specified, the parameters are set as follows: $L = 200$ m, $K = 5$, $r = 40$ m. The power consumption in receiving, transmitting and sleep is 13.5 mW, 24.75 mW, 15 μW, respectively [3]. Each data-gathering period is 1 second long. In each data-gathering period, every node generates 512 bits data. The initial energy of a node is 1 J and radio bandwidth is 115 kbps.

To investigate the energy efficiency of DTDGS, we compare DTDGS with S-MAC [3] and AIMRP [4] in terms of packet delivery ratio and network lifetime. We assume that S-MAC is coupled with a routing protocol that imposes no additional protocol overhead, and routes packets to the sink in the least number of hops.

Figure 4 illustrates the comparison of packet delivery ratio among three schemes when the number of coronas increases. Packet delivery ratio is defined as the ratio of data packets received by the sink to those generated by all sensor nodes. The node density $\lambda = 0.004$. Figure 4 shows the packet delivery ratio in DTDGS is higher than those in S-MAC and AIMRP. It testifies that DTDGS can avoid transmission collision.

Figure 5 depicts the network lifetime versus the total number of nodes for three schemes, where network lifetime is defined as the time elapsed utill 1 percent of the nodes in the network run out of energy. As shown in Fig. 5, DTDGS improves the network lifetime over SMAC and AIMRP. This is because DTDGS uses TDMA-based scheduling to remove the idle listening and avoid transmission collisions. Besides, the balanced energy consumption among nodes in the same corona can be achieved in DTDGS. The simulation results indicate high energy efficiency of DTDGS.

6. Conclusion

In WSNs, every node can be in active (for receiving, transmission activities), idle listening and sleep modes. In idle listening mode, the nodes consume almost the same amount of energy as in active mode, while in sleep mode, the nodes shutdown the radio to save energy. In the DTDGS scheme, we take advantage of TDMA-based scheduling to remove the idle listening because every node has a fixed time slot for transmitting and receiving. Thus, significant energy savings can be accomplished by this scheme that can let the sensors' transceivers remain in sleep mode as long as possible. In DTDGS, each node builds its own schedule and selects its next-hop forwarding node according to the information gathered from neighbor nodes. Different from the centralized algorithm, the DTDGS does not need a centralized gateway to assign the transmission time slots and compute the route for each node. In DTDGS, the schedule set-up phase is needed for creating the schedule. This procedure takes place when the network is initialized, and whenever sensor node failures are expected or experienced. Therefore, the DTDGS scheme is energy efficient and highly scalable, thus apt for large-scale data-gathering WSNs.

References

[1] Y. Wang and I. Henning, "A deterministic distributed TDMA scheduling algorithm for wireless sensor networks," Intl Conference on Wireless Communications, Networking and Mobile Computing, pp.2759–2762, 2007.

[2] N.A. Pantazis, D.J. Vergados, D.D. Vergados, and C. Douligeris, "Energy efficiency in wireless sensor networks using sleep mode TDMA scheduling," Ad Hoc Networks, vol.7, no.2, pp.322–343, March 2009.

[3] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," Proc. IEEE INFOCOM, pp.1567–1576, 2002.

[4] S. Kulkarni, A. Iyer, and C. Rosenberg, "An address-light, integrated MAC and routing protocol for wireless sensor networks," IEEE/ACM Trans. Netw., vol.14, no.4, pp.793–806, Aug. 2006.