RC-MAC: A Receiver-Centric Medium Access Control Protocol for Wireless Sensor Networks

Pei Huang, Chen Wang, Li Xiao
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI, USA
Email: {huangpe3, wangchen, lxiao}@cse.msu.edu

Hongyang Chen
Institute of Industrial Science
The University of Tokyo
Tokyo, Japan
Email: hongyang@mcl.iis.u-tokyo.ac.jp

***Abstract*—Wireless sensor networks usually operate under light traffic loads. However, when an event is detected, a large volume of data may be generated and delivered to the sink. The demand for simultaneous data transmission may cause severe channel collision and thus decrease communication throughput in contention-based medium access control (MAC) protocols. In this paper, we introduce a novel receiver-centric data transmission paradigm, which takes advantage of the tree structure that is naturally formed in data collection of a sensor network to assist scheduling of channel access. On the tree structure, a receiver is able to coordinate its multiple senders' channel access so as to reduce channel contention and consequently improve communication throughput. The protocol seamlessly integrates scheduling with contention-based medium access control. In addition, to ensure reliable data transmission, we propose a sequence-based lost packet recovery scheme in a hop-by-hop recovery pattern, which could further improve communication throughput by reducing control overhead. We present the performance of our receiver-centric MAC protocol through measurements of an implementation in TinyOS on TelosB motes and extend the evaluation through ns-2 simulations. Compared with B-MAC and RI-MAC, we show the benefits of improving throughput and fairness through receiver-centric scheduling under heavy traffic loads.**

I. Introduction

A wireless sensor network (WSN) usually consists of a large number of small sensing devices that are powered by batteries, equipped with less capable processors and limited memory. Reporting sensed data to a base station (usually referred to as the *sink*) is the primary function of a sensor network. While sensor nodes normally operate under light traffic loads, they may be suddenly activated by abrupt events such as enemy attack or fire detection. In such cases, a large volume of data may be generated and transmitted to the sink within a short period of time, which demands a high throughput communication channel to transmit the bursty traffic in a timely fashion. However, the severe channel collision, due to the simultaneous needs of data transmission together with the lossy nature of wireless channel, makes it a challenging task to achieve high throughput and reliable transmission in such an occasion.

Schedule-based TDMA (Time Division Multiple Access) schemes may be a natural choice to address the intensive collision problem. However, TDMA-based MAC protocols usually introduce extra overhead for synchronization, time slot assignment, and schedule maintenance. Moreover, idle time slots reduce the channel utilization of TDMA since not all nodes have pending data to send. Z-MAC [1] addresses the underutilization problem by allowing nonowners of a time slot to contend for the slot with less access probability than that of the owner. Consequently, the denser the network is, the closer to CSMA (Carrier Sense Multiple Access) the Z-MAC is. The reason is that in a dense network a large proportion of time slots are assigned to neighboring nodes that have no data to send and nodes on the delivery paths have to contend for these time slots.

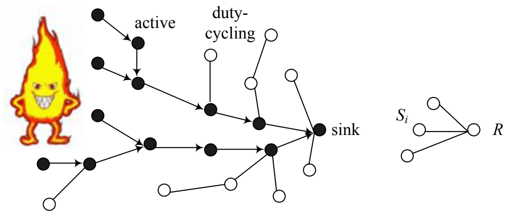

Fig. 1: RC-MAC utilizes the tree structure to assist channel access scheduling

Comparing with rigid TDMA schemes, contention-based CSMA mechanisms are more flexible and simple. No synchronization or topology information is required, and nodes leave and join in network without incurring any extra operation. Therefore, they can be easily deployed and flexibly adapted to a dynamic sensor network topology. However, the contention nature of CSMA may intensify channel collision and this motivates us to integrate TDMA's collision-free characteristic into CSMA while retaining CSMA's flexibility and low overhead.

An observation of traffic in WSNs is that when an event is detected, many nodes may generate reports and send them to the sink. These messages can be combined or aggregated at some intermediate nodes and in general, they naturally form a hierarchical tree structure as shown in Figure 1. In the hierarchical tree structure, each intermediate node receives packets from multiple senders and forwards packets to its own parent at a higher layer. A basic unit consists of one parent (receiver) and multiple children (senders). The central position of the parent makes it an ideal candidate to manage channel access scheduling, which can reduce channel contentions from multiple children to only one node. In this paper, we show that many benefits can be gained by switching the scheduling function to the receiver side.

As acknowledgement (ACK) can be a nontrivial overhead, to ensure reliable data transmission, we propose a sequence-based lost packet recovery scheme that can be used to reduce the control overhead. While sequence-based mechanism has been successfully applied in the sliding-window algorithm of TCP protocol, we found that the sequence-based mechanism, originally designed for end-to-end protocols, is not robust and incurs extra overhead when directly ported to hop-by-hop recovery, which is the preferred retransmission mechanism in WSNs. Therefore, how to address the limitations of sequence-based mechanism in a hop-by-hop recovery pattern will be detailed in the coming discussion.

The major contributions of this work are as follows:

- We propose to use tree structure that is naturally formed in data collection of sensor networks to assist channel access scheduling. The scheduling is traffic adaptive and robust to topology changes.
- We shift the scheduling function to the receiver side so that it avoids collision within one basic unit and achieves high throughput. Moreover, with our proposed blocking mechanism, collisions among units are also mitigated.
- The scheduling takes into account the different bandwidth demands of nodes on a collection tree and assign them different channel access opportunities.
- We dynamically adjust the rounds of scheduling of a unit according to the remaining buffer size, hence no unit can occupy the channel exclusively. Therefore, our proposed receiver-centric MAC protocol can ensure good fairness.
- We apply sequence-based mechanism in a hop-by-hop recovery pattern which further improves communication throughput by reducing control overhead.

In the following discussions, we present the design, implementation, and performance evaluation of our receiver-centric MAC protocol (RC-MAC). Section II provides an overview of the basic concept of RC-MAC and outlines design challenges. Section III details the channel access scheduling and Section IV discusses the reliable data transmission. Performance evaluation is given in Section V and related work is summarized in Section VI. We finally conclude this paper in Section VII.

II. Overview

In this section we present the basic concept of RC-MAC: scheduling by utilizing the underlying tree structure. We also discuss the main goals and challenges of integrating scheduling into CSMA.

A. Basic Concept

Common applications [2] [3] in WSNs determine that routing protocols in WSNs typically form a collection tree. For example, the default routing protocol in TinyOS 2.x is the Collection Tree Protocol (CTP) [4], in which one or more nodes in the network declare themselves as sink nodes and all other nodes in the network recursively form routing trees using ETX (expected transmission count) [5] [6] as the routing metric [7]. Taking advantage of the underlying tree structure that exists specifically in WSNs, we can approach TDMA's collision-free environment while do not introduce complex operations and extra overhead.

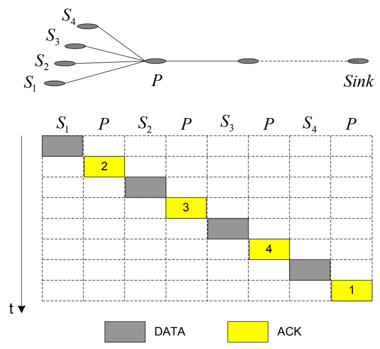

Fig. 2: RC-MAC schedules time slots to different children based on overhearing

Figure 2 shows an overview of how to utilize the tree structure in RC-MAC. A receiver schedules its children's packet transmissions by reusing the ACK. Due to the broadcast nature, the ACK sent to one of its children can be overheard by all of its children. Therefore, we can schedule children's packet transmissions by piggybacking a scheduling message, which is the ID of the child that can send a packet the next time, to an ACK as shown in Figure 2. Overhearing an ACK containing a scheduled node ID, a child holds its packet if the scheduled node ID is not equal to itself, or starts to transmit if the scheduled node ID matches its own. As a result, channel collision can be significantly reduced since only the scheduled child starts to transmit while all other siblings hold their transmissions. RC-MAC operates as "pulling" data rather than traditional pattern of "pushing" data. This benefits from the special many-to-one data collection traffic pattern in WSNs. The failure of a child node is handled by the scheduling method described in Section III-B and the failure of a parent node is handled by the CTP [4] where children switch to another parent according to updated ETX.

B. Design Goals and Challenges

Our main design goal is to reduce collision and improve communication throughput under heavy traffic loads by utilizing the underlying tree structure in WSNs. However, we do not want to introduce extra overhead for time slot assignment because it is not really traffic adaptive. Each node has to be assigned a time slot but not all of them have data to send in a particular period. With only a little information available, we face two-step design challenges.

First, due to the unbalanced tree structure, different nodes have different bandwidth demands and thus we should assign nodes different chances for channel access instead of simple round robin polling. We also need to address topology changes as nodes may want to join or leave the scheduling or dynamically change their parents as link quality fluctuates.

Second, having solved collision problem in a basic unit, challenges come out when you put it into a multi-hop scenario where contentions among units still exist. We cannot let one basic unit hold the channel exclusively and we do not want the scheduling to be interrupted by contentions among units.

III. Design Details

A. Adapt to Low Duty Cycle

Our proposed RC-MAC is designed to be effective under heavy traffic loads when events of interest are detected. Their occurrence usually results in long bursts of data packets. Hence, they typically represent the overall network traffic although they occur rarely [7]. Nevertheless, in most of the time, network generates very little or no data traffic. In order to improve energy efficiency, RC-MAC should be able to adapt to the widely adopted duty cycling [8] [9] technique in WSNs. In this technique, nodes turn their radio on only periodically, alternating between active and sleeping states.

Under light traffic loads, we design RC-MAC to adopt the paradigm of recent proposed RI-MAC [10], which is shown to be better than prior LPL (low power listening) based solutions such as B-MAC [9] and X-MAC [11]. In this paradigm, senders remain active and wait silently. When a node wakes up, it broadcasts a beacon to notify neighboring nodes that it is ready to receive data. Upon receiving the beacon, multiple senders contend for channel access. Different from RI-MAC where multiple senders transmit data immediately upon receiving a beacon, RC-MAC requires initial backoff here because RI-MAC's fairness is poor as we noticed in experiments that the capture effect of radio may provide nodes with higher transmission power a greater chance to transmit successfully and continuously. In addition, RC-MAC does not rebroadcast a beacon with increased backoff window size when detects a collision because the extra overhead does not bring any throughput gain as the contention is synchronized. Instead, senders in RC-MAC will retry with binary exponential backoff (BEB) if an ACK times out. The receiver is definitely awake if it has received some data and thus becomes a sender. In an extreme case where packets sent to a receiver were all collided and the receiver goes back to sleep, senders stop retrying if the retrying count reaches a predefined limit. They wait for the next time the receiver wakes up and broadcasts the beacon.

Any other duty-cycling MAC protocols such as B-MAC [9] and X-MAC [11] can also be adopted under light traffic loads. In either B-MAC [9] or X-MAC [11], senders will also stay awake. Therefore, all nodes on the delivery paths will finally switch from duty-cycling mode to active mode. The fact is the basis that CSMA can be used in RC-MAC. Currently, we adopt the paradigm of RI-MAC with slight modifications to accommodate the light traffic. When senders hear a beacon or an ACK from a receiver, they set the flag of the receiver as active. They only set the flag as dormant when the retrying count reaches a predefined limit or they finish delivery and go back to the duty-cycling mode. A node, on the other hand, stays awake if it has pending data to send. The design of RC-MAC for heavy traffic loads follows.

B. Channel Access Scheduling

We have demonstrated that there exists an underlying tree structure in the network as illustrated in Figure 1. As nodes operate under low duty cycle, a node broadcasts a beacon when it wakes up. Children nodes that have pending data towards it

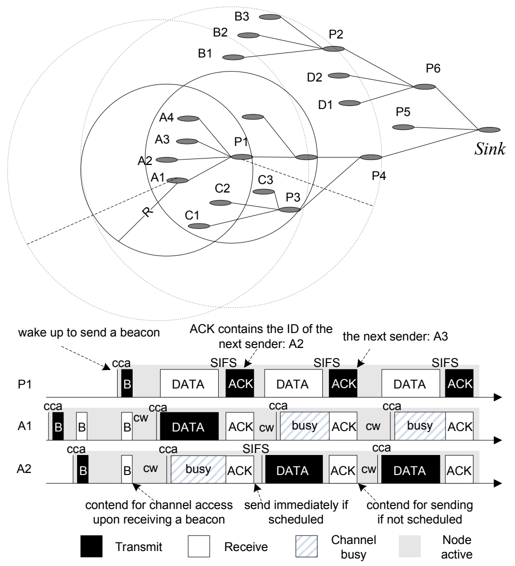
B3
B2
B1
P2
D2
D1
P6
P5
Sink
A4
A3
A2
A1
P1
C2
C3
P3
C1
P4
R
wake up to send a beacon
ACK contains the ID of the next sender: A2
the next sender: A3
cca
SIFS
P1
B
DATA
ACK
A1
cw
busy
A2
contend for channel access upon receiving a beacon
send immediately if scheduled
contend for sending if not scheduled
Transmit
Receive
Channel busy
Node active

Fig. 3: A simple example of RC-MAC. Use circle to represent interference among units for simple presentation only (solid circle: communication range; dotted circle: interference range).

start to contend for sending. Upon receiving a data packet, the parent node responds an ACK that contains the ID of the next sender. The scheduled child node will transmit immediately after it receives an ACK from its parent. On the contrary, neighboring nodes will refrain from sending for a random period of time and then start for contention if the receiver flag is set as active. Designing different contention window for the random backoff can further distinguish unscheduled sibling nodes from nonchildren nodes of the parent. Upon receiving an ACK, nonscheduled siblings perform in-group contention with contention window (CW) of $(SIFS + CCA,\ T_1]$, nonchildren nodes perform ongoing-transmission contention with CW $(T_1,\ T_2]$, and nodes within interference range use network allocation vector (NAV) as virtual carrier to avoid the hidden terminal problem when they sense the transmission of ACK. The EIFS (extended interframe space) is thus set to be long enough for an ACK-DATA-ACK round. Since the parent knows the number of its children, T_1 can be set optimally to handle contentions among children by analytical techniques [1], and T_2 is set as the commonly used BEB value.

Taking Figure 3 as an example, A_1 wins channel access and transmits a packet to P_1 upon receiving a beacon from P_1. During the DATA transmission period, nodes in unit C cannot transmit because they are in the interference range of unit A and can sense the busy channel. Upon receiving a packet from A_1, P_1 immediately responds an ACK to A_1 after a delay of SIFS (short interframe space). Supposing A_2 is the next child specified to transmit in the ACK, it will immediately start the CCA (clear channel assessment) outlier algorithm, which is introduced in B-MAC [9] for accurately determining whether the channel is clear or not. If the channel is clear, A_2 sends a packet. If A_2 does not have data to send or cannot send due to busy channel or process delay of chips, the backoff timers of siblings will expire and they will contend for channel access. If P_1 cannot hear from A_2 for 3 times, it removes A_2 from the schedule list assuming A_2 has no data to send. P_1 reduces the original tree structure to a smaller schedule list to achieve the goal that only nodes with traffic are assigned time slots, which yields better channel utilization than TDMA. A_2 can still join in the contention for sending if it has data to send later. If there is no such a chance, it waits for the next time when the parent reinitializes the scheduling list as shown below.

C. Dynamic Adjustment for Fairness Among Units

Normally, P_1 will keep scheduling its children to transmit and dominate the channel. For fairness consideration, after several rounds of scheduling, we need to penalize unit A to give the parent P_1 and neighboring units (e.g., unit C) the chance to win channel access. The number of rounds is determined dynamically by the receiver's remaining buffer size. For instance, suppose a node's buffer size is 50 packets and currently the queue size is 10 packets, then the remaining buffer size is 40 packets. We allow half of the remaining buffer size to be used in a continuous scheduling. Therefore, the node responds an ACK with -1 instead of the ID of the next sender to terminate the scheduling once it has received $0.5 \times 40 = 20$ packets from its children. The dynamic adjustment prevents unbalanced packet accumulation. If one unit is easier to win channel access than other units, the parent's buffer size will shrink quickly. As a result, the unit is unlikely to keep scheduling, and the dynamic adjustment prevents any unit from occupying the channel exclusively.

Hearing this type of ACK, neighboring nodes will contend for channel access immediately since no data will follow this type of ACK. The parent node (i.e., P_1) also participates in the contention trying to relay packets to its own parent. All the children of P_1, however, will refrain from contention for a certain period of time as punishment. This reduces the overall contention and balances the traffic. When the next time P_1 starts scheduling, the schedule list is recovered to be the original children set to accommodate traffic changes.

D. Account for Different Bandwidth Demands

Scheduling children in a simple round robin manner seems to be fair in a basic unit scope. However, it is unfair in a global view. For example, in Figure 3, P_2 should be given more opportunities for sending than D_1 and D_2 because in addition to its own data, it also collects data from B_1 to B_3. Let b_u denote the total bandwidth demand of node u and d_u represent the data rate generated by node u. The total bandwidth demand of node u is equal to the total bandwidth demand of u's subtree plus the data rate of itself, which is $b_u = \sum_{c_u^i \in C_u} b_{c_u^i} + d_u$ where C_u is the children set of node u and c_u^i is the i-th child of node u. The probability that node u will be chosen as the next sender by its parent p_u is then equal to $b_u / \sum_{c_{p_u}^i \in C_{p_u}} b_{c_{p_u}^i}$. In this way, heavy traffic nodes get more channel access opportunities and thus the fairness is improved. The similar idea is also used in [12] for time slot assignment. The bandwidth demand can be piggybacked on the data packets when changed. The information that there is no data to send any more can also be piggybacked so that a sender will be removed from the schedule list.

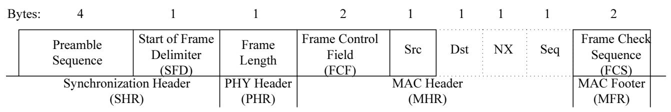

Fig. 4: The format of a RC-MAC beacon/ACK frame for an IEEE 802.15.4 radio

E. Collision Among Units

To maximize the benefits of receiver-centric scheduling, we need to minimize the interferences among units; otherwise, it may degrade to CSMA. Due to the hidden terminal problem, some nodes may be unable to sense the transmission of other units and trigger their own receiver-centric scheduling. To avoid interrupting ongoing scheduling of neighboring units, a receiver changes the value of the next sender ID to -2 when it responds an ACK but detects busy channel. Children of this unit will backoff several rounds of data transmission before retry contention, while nonchildren nodes regard it as a chance to win channel access since no immediate data will follow the ACK with -2.

In addition to the general rule, there are time priority and capture effect exception for maintaining the ongoing scheduling. For example, as shown in Figure 3, although A_1 wins channel access, B_3 is not in the interference range of A_1 and it may transmit a packet to P_2, which will trigger the receiver-centric scheduling within unit B. If the reception of P_1 is corrupted by the transmission of B_3, unit A will not enter in the receiver-centric scheduling because P_1 will not respond an ACK. If P_1 receives A_1's packet successfully, it assumes that unit B will quit since unit A holds the channel earlier and thus P_1 should start scheduling normally (i.e., time priority). The reception of P_2 may be interfered by the ACK of P_1. If the reception of P_2 is corrupted, the receiver-centric scheduling of unit B will not be triggered, and the children will keep contending. If P_2 can capture the data packet, P_2 will also start scheduling since they are far away enough (i.e., capture effect exception).

Note that no contention exists in a basic unit. Among units, only those out of interference range can coexist. The scheduling considerably reduces the probability of collision compared to a scenario where all sensor nodes in the same interference range keep contending for packet transmission, and also improves channel utilization by allowing compatible units transmit simultaneously.

F. Beacon/ACK Frame

A beacon frame is used when a node wakes up. The aim is to notify neighboring nodes that it is ready to receive packets. As a result, it must contain a *Src* (source ID) field. In addition to this, an ACK must also contain a *Dst* (destination ID) field and a *NX* field which is used to specify the next sender. An optional *Seq* field is introduced in the following sequence-based lost packet recovery scheme. The beacon and ACK frame can be combined together and a node can distinguish them according to the length of the frame. We comply with the IEEE 802.15.4 frame format to design our beacon/ACK frame format, which is illustrated in Figure 4.

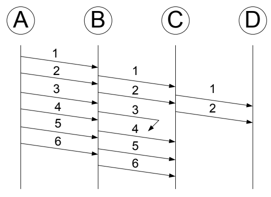

Fig. 5: In TCP protocol, node B stops forwarding packet 4, 5, and 6 when packet 3 is lost

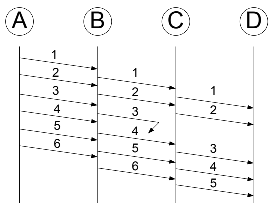

Fig. 6: In RC-MAC, node B continues to forward packet 4, 5, and 6 even when packet 3 is lost

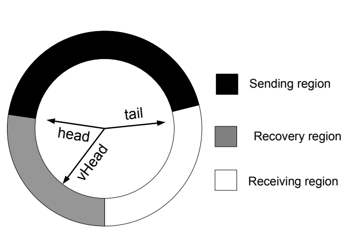

Fig. 7: RC-MAC divides packet buffers into three regions: sending region, recovery region, and receiving region

IV. Reliable data transmission of the receiver-centric MAC protocol

Retransmission mechanism is widely used to ensure reliable data transmission in WSNs. The time-out mechanism is a basic approach to retransmit lost packets. In this mechanism, a sender will wait for an ACK after it sends out a packet to a receiver, and retransmit the packet if the ACK is not received within a certain period of time. The lost packet may be retransmitted multiple times up to a certain threshold. The time-out mechanism is simple and can be easily implemented. However, we have to pay the penalty of large control overhead. Take CC2420 radio as an example, for a packet with payload of 28 bytes, there are 19 bytes headers, thus in total 47 bytes by default in TinyOS. An ACK, on the other hand, is 11 bytes, about one quarter of the DATA frame. Aggregating small packets into a long message has the drawback of high cost of retransmission if only a few bits have been corrupted. Therefore, in the second part of RC-MAC design, several small packets are aggregated as a group with one ACK, and a sequence-based mechanism is adopted within a group to detect and retransmit lost packets. In this design, the number of ACKs is reduced and only the lost packets in a group, instead of the whole group, will be retransmitted.

To detect lost packets between a sender and a receiver, the sequence-based mechanism labels the packets sent in a group with continuous sequences, and lost packets can be detected by the receiver if a discontinued sequence is received. We shift the retransmission decision to the receiver side because the sender may lose some overhearing packets and incorrectly thought that some packets with missing sequences are lost.

The sequence-based mechanism, originally designed for the sliding window algorithm of the end-to-end TCP protocol, works between a pair of end users to ensure a highly reliable, strictly in-order packet transmission. Different from the TCP protocol, the RC-MAC applies the sequence-based mechanism in a hop-by-hop recovery manner to achieve reliable data transmission from multiple sources to a sink. This difference leads to different design principles and implementation details.

A. Streaming Data Transmission from Multiple Sources to A Sink

As we discussed before, the RC-MAC is designed to maximize throughput from multiple sources to a sink. Based on this principle, we design the RC-MAC to stream packet transmission such that it will not be interrupted by the retransmission of lost packets. We first show that continuous packet forwarding can be interrupted by the sequence-based mechanism directly ported from the TCP protocol. After that, we discuss how the continuous data stream is maintained in RC-MAC.

The sequence-based mechanism relies on strictly in-order sequence to detect lost packets, which may incur extra overhead and interrupt the packet forwarding stream. As illustrated in Figure 5, when the source sends out packets to the sink through multihop forwarding, the source labels packets with continuously increased sequences. When node C receives packets 4 to 6 while losing packet 3, it will issue a request to node B to resend packet 3. At this moment, packets 4 to 6 have to be held by node C and cannot be sent to node D. Otherwise, node D will also detect that packet 3 is missing and request node C to retransmit. Here, all the subsequent nodes have to wait until node C recovers the single lost packet 3. Therefore, the data forwarding stream is interrupted by retransmission of lost packets. The situation may become worse if some packets cannot be recovered due to buffer overflow. Those lost packets will continue to be detected by intermediate forwarding nodes and incur frequent lost requests for unrecoverable packets.

The sequence-based mechanism cannot maintain a continuous data forwarding stream because it relies on strict continuous sequences globally maintained between the source and the sink. Therefore, an interruption at any intermediate node will stop data forwarding of the entire path. To solve this problem, we add a localized numbering mechanism to relabel each packet at each hop. In this mechanism, when a node sends packets, it re-labels packets with continuously increased sequences maintained in the local variable. An example is shown in Figure 6, node C detects lost packet 3 based on discontinued sequences. However, node C can still continue forwarding packets 4 and 5 to node D since all packets are re-labeled by node C when sent out. The packet forwarding between nodes B and C and the packet forwarding between nodes C and D are isolated by the sequence renumbering. Therefore, packet interruption at one hop will not affect the continuous forwarding of the entire path. When a lost packet is received, it will be inserted to the right position of the queue according to the original global sequence or sent out before any other packet to catch up with the global sequence.

B. Request Lost Packets

Each time a node sends 8 packets in a group, and one byte (i.e., *Seq* field in Figure 4) is attached to the ACK for requesting lost packets. Each bit is the representative of a packet in the group: '1' implies received, '0' means lost.

Upon receiving the lost packet sequences, we slightly modify the queue management with an extra virtual head to recover lost packets with O(1) time complexity. In RC-MAC, the queue is divided into three regions: the sending region, the receiving region, and the recovery region as shown in Figure 7. The sending region contains all the packets that wait to send, the recovery region contains the 8 packets that have been sent, and the receiving region contains empty buffers that wait for new packets. The sending region works as a normal queue, which sends out packets at the *head* and receives packets at the *tail*. However, when a packet is sent out, it will be temporarily moved from the sending region to the recovery region, which might be used to recover lost packets as follows.

When the sender is notified by the receiver with missing sequences, the lost packets can be recovered from the recovery region. To achieve this, we use an extra pointer named *vHead*, which can be temporarily pointed to the buffer containing the lost packets. When the lost packets are resent, the normal *head* will continue to be used for packet forwarding (the lost packets and the new packets are sent together in a group). Here *vHead* is used to lookup and resend the lost packets. In RC-MAC, looking up a lost packet in the recovery region can be finished in O(1) time because the sequence is relabeled when sent out, the sequence number s assigned to a packet is correlated with the index i of the buffer containing the packet. We have $i = s \bmod N$, where N is the recovery buffer size.

V. Performance Evaluation

We first evaluated RC-MAC in an implementation in TinyOS on TelosB motes and then extended the evaluation in ns-2 network simulator. We compared our RC-MAC with B-MAC [9] and RI-MAC [10]. Assuming all nodes are active, we have a variant of B-MAC where the LPL is disabled. Then B-MAC works like a pure CSMA and can achieve high throughput without the preamble transmission. This comparison allows us to demonstrate the benefits of coordinating channel access among neighboring nodes. We have the same philosophy as RI-MAC that shifts responsibilities to the receiver side, but RI-MAC is contention-based and adopts random backoff to mitigate collisions. On the contrary, RC-MAC is schedule-based, ensuring no collision within a unit. Thus we compared our RC-MAC with RI-MAC to demonstrate the advantages of utilizing the tree structure in a WSN.

A. Experimental TinyOS Evaluation

RI-MAC [10] was implemented under the unified power management architecture (UPMA) framework [13] [14] in TinyOS, where B-MAC [9] is also provided. To compare RC-MAC with them, we also implemented our RC-MAC under the UPMA framework.

To adapt to heavy traffic loads, we made a minor modification on RI-MAC. In the original implementation of RI-MAC, a node broadcasts a base beacon with no backoff window field when it wakes up. Then all senders with pending DATA frames transmit immediately. This design is optimized for a typical sensor network where there is light or no traffic most of the time. Under heavy traffic loads, however, due to the capture effect of CC2420 radio, we found that its fairness is extremely poor since the received signal strength from different senders is different. Consequently, some nodes' transmissions are more likely to be captured while others have a lower chance to be received. Therefore, we included an initial backoff value in the base beacon.

CC2420 includes hardware support for transmitting ACK, but we are unable to modify the frame format. Therefore, same as in RI-MAC, RC-MAC uses a frame with only the CC2420 header as the ACK frame. The length is thus 19 bytes, although we can decrease it to 14 bytes as described in Section III-F. For fair comparison, B-MAC also adopted this MAC layer ACK (MACK) when LPL was disabled because in both RI-MAC and RC-MAC, several milliseconds are used to load the MAC layer ACK into CC2420's TX buffer. This variant is named as BMAC-MACK and the original B-MAC is called BMAC-LPL-SACK in the following discussions (ACK is hardware supported but is issued by a software command *SACK.strobe()*, hence SACK).

The payload of a DATA frame is 28 bytes. Adding the headers, the length is 47 bytes in total, which results in an airtime of about 1.5 ms with 250 kb/s data rate of CC2420. The preamble length is set to 20 ms in BMAC-LPL-SACK and the beacon timer in RI-MAC is set to 1 s to minimize the control overhead while achieving good fairness. The throughput is measured every minute and averaged over 10 minutes.

To demonstrate the fairness, we use the metric *fairness index* defined in [15] [16]. The aim of fairness based on throughput is to provide the same throughput to all sources. Consider a system with n flows, each receiving a throughput of T_i, the fairness index is given by

$$F = \frac{\{\sum_{i=1}^{n} T_i\}^2}{n\{\sum_{i=1}^{n} T_i^2\}}$$

Experimental results in one-hop topology. We first evaluated BMAC-LPL-SACK, BMAC-MACK, RI-MAC, and RC-MAC in a one-hop topology scenario where n nodes are placed equidistant from a receiver in a circle. Each node transmits as quickly as possible. Upon receiving an ACK, a node immediately generates another packet, or if ACK times out, *Resend* command is called. This is to measure the achievable throughput of different MAC protocols under different levels of contention in a basic unit. We forced radio on so that B-MAC can disable the LPL. The results are shown in Figure 8. A source in the figure is a node that generates packets.

When there is only one sender, the sender is always specified as the next sender in RC-MAC. It transmits a DATA frame without initial backoff and thus increases the channel utilization. On the contrary, BMAC-MACK and RI-MAC both send a DATA frame after a random initial backoff. Their difference is that the RI-MAC needs to broadcast beacon periodically and thus BMAC-MACK achieves higher throughput. Moreover, in RI-MAC, if the receiver gives up beaconing due to consecutive collisions, all senders have to wait for the next time the beacon timer of the receiver expires. Nevertheless, RI-MAC indeed improves on B-MAC if LPL is needed in B-MAC.

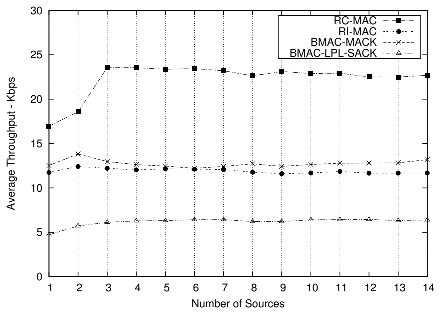

Fig. 8: Data throughput comparison in the one-hop TelosB topology

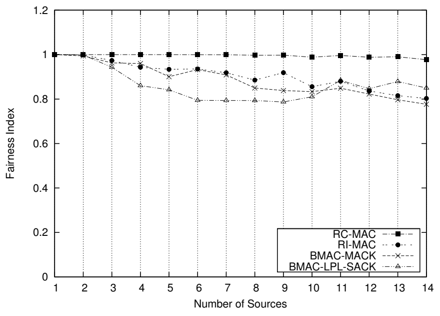

Fig. 9: Fairness index comparison in the one-hop TelosB topology

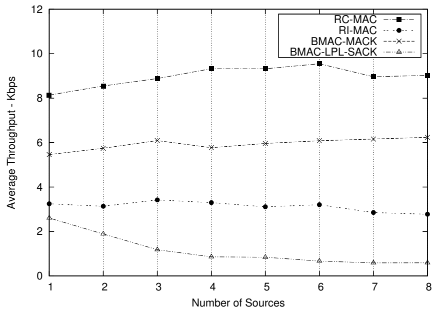

Fig. 10: Data throughput comparison in the two-hop TelosB topology

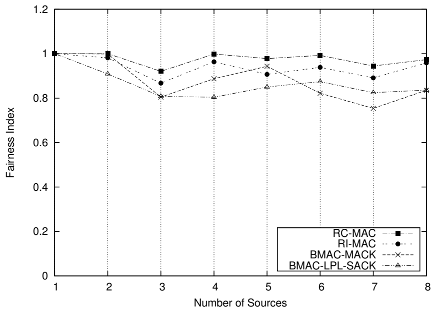

Fig. 11: Fairness index comparison in the two-hop TelosB topology

The throughput is limited by process delay when only one sender is presented. It usually takes several milliseconds to load a packet to the TX FIFO of CC2420. Therefore, when more than one sender is deployed, the specified next sender has already loaded the DATA into the TX FIFO and thus can transmit immediately. The throughput of RC-MAC soars high to the maximum achievable throughput when 3 senders are deployed. The throughput is now upper bounded by the ACK process delay of the receiver. The throughput of B-MAC and RI-MAC increases slightly and drops a little. The reason is that all packets have to go through the initial backoff or congestion backoff, and packets may collide with each other when more sources are contending.

The fairness comparison is illustrated in Figure 9. The high performance of RC-MAC comes from the scheduling that each sender has equal chance to transmit without interference as coordinated by the receiver. On the contrary, the fairness indices of B-MAC and RI-MAC drop to 0.8 as the number of senders increases. The random initial backoff, congestion and collision leave nodes unfair chances to deliver packets.

Experimental results in two-hop topology. To explore possible problems in multi-hop relay, we added a manual routing in TinyOS implementation. The number of children of the sink is increased from 1 to 4, and the number of children for each child is increased from 1 to 2. The leaf nodes are sources. As in the one-hop topology, all sources always have data to send. The results are presented in Figure 10.

The throughput of RC-MAC and BMAC-MACK is reduced by a little more than half compared with the one-hop topology scenario because a packet needs to be relayed by two hops to arrive at the sink. Compared with BMAC-MACK, the results demonstrate that coordination among senders helps to save time for backoff and reduce collisions as well. The throughput of RI-MAC is decreased by two thirds in the two-hop scenario. The reasons are as follows. In RI-MAC, a node sends an ACK beacon when it receives a DATA frame, and this ACK beacon invites more DATA frames to it. Only when consecutive collisions make it give up beaconing or no more data is destined to it, it starts to send its own data. In turn, it needs to wait for its own intended receiver's beacon. In our RC-MAC, we have dynamic adjustment according to the remaining buffer size. When a node accumulates enough data, it will penalize the contention of its children and starts its own data transmission. No unit will hold the channel exclusively.

Fairness comparison illustrated in Figure 11 shows that although RI-MAC delivers fewer packets, its fairness is better than B-MAC. On the one hand, the lower delivery ratio contributes to the fairness. On the other hand, all senders hold their packets and wait for the intended receiver's beacon to initiate data transmission. The synchronized contention in general assigns more equal chance to senders than BMAC-MACK where nodes perform congestion backoff.

B. Simulation Evaluation

We further extended the performance evaluation in ns-2. To match the real implementation, we drew parameters from the CC2420 datasheet [17]. The RSSI value is averaged over 8 sample periods (128 μs), hence we use this time as the CCA check delay. The hardware supported ACK is transmitted 12 symbol periods (192 μs) after the last symbol of the incoming

TABLE I: The default simulation parameters

Initial backoff window	31×10
Binary exponential backoff window	$(31 \sim 255) \times 10$
CCA check delay	128 μs
SIFS	192 μs
Slot time	30.5 μs
Process delay	6 ms
Transmission range	135 m
Interference range	275 m
Communication bandwidth	250 Kbps

frame. We use this time as the value of SIFS. TinyOS provides a 32 KHz timer, hence the slot time is $1s/32768 = 30.5\ \mu s$. We also add 6 ms process delay as observed in experiments to the simulator. It compensates for writing/reading TX/RX FIFO of CC2420. The transmission/interference ranges are set as reported in [18]. Table I summarizes the default settings of the simulation parameters.

In simulations, we implemented our dynamic adjustment in RI-MAC so that a receiver will stop requesting for data according to its remaining buffer size. For B-MAC, we disabled the LPL function. We skip simulation results in the one-hop topology because of space constraint. The results are consistent with the experimental results.

Simulation results in two-hop topology. We created two-hop topologies where hidden terminals may degrade the performance of different protocols. In the topology, the sink has three children and each child receives data from multiple sources. We increased the number of sources from 2 to 6 for each of the three intermediate nodes. We also adjusted the distance between nodes and averaged the results on five graphs. Figure 12 presents the data throughput comparison when hidden terminals exist.

With our blocking mechanism described in Section III-E, RC-MAC is robust to hidden terminals. A hidden terminal may trigger receiver-centric scheduling in a unit and interfere with another unit. However, the receiver will penalize the contention of its children once detects a collision and consequently quits the scheduling. Only compatible units can coexist and the interference is mitigated. Since there is no contention within a basic unit, RC-MAC obtains the highest throughput.

RI-MAC has lower throughput than B-MAC because sources can only contend for sending when requested by receivers in RI-MAC. As a result, when two units can transmit simultaneously, only one unit is active while the senders of the other unit are waiting for beacon. Furthermore, when a child of the sink receives enough packets, it also needs to wait for the sink's beacon to initiate data transmission.

Although B-MAC delivers more packets than RI-MAC, its fairness is worse than RI-MAC as shown in Figure 13. Nodes with fewer neighbors get a greater chance to send when they all keep contending in B-MAC. Our dynamic adjustment ensures fairness in that although the boundary nodes have a greater chance to grab the channel, the receiver will penalize them accordingly.

Simulation results in multi-hop topology. Having verified the effectiveness of receiver-centric scheduling, we finally constructed a network with 50 nodes that are randomly scattered in a $1000 \times 1000\ m^2$ field. We fixed the number of sources to be 20 and increased the data rate from 2 pps (packets per second) to 40 pps. We averaged the results over 10 graphs. Figure 14 shows the throughput performance under the multi-hop topology of 50 nodes.

When data rate is low, CSMA-based B-MAC is better than our schedule-based RC-MAC. When data rate is low, a child that is scheduled to transmit in RC-MAC may not have data to send. Other siblings have to hold their packets and contend for transmission only when their backoff timers expire. If all children have no data to send, the channel is reserved but nothing has been sent until neighboring nodes begin to contend. Therefore, scheduling occupies the channel but no data is transmitted. It is no better than just letting nodes with pending data to send contend for transmission. In RI-MAC, if no data is received after beaconing, this receiver stops beaconing and resets the beacon timer. When a node has data to send, it must wait for the receiver's beacon timer to expire while in B-MAC it can transmit immediately.

As data rate increases, the performance of B-MAC degrades due to collisions. The performance of RC-MAC also has a slight drop because nodes always have data to send and hidden terminals may interrupt neighboring units' receiver-centric scheduling. The hidden terminal problem also affects RI-MAC, but since all nodes wait for beacons, the interference is comparably mild.

VI. Related work

Many studies have been done to provide low duty cycle MAC protocols [9] [11] [10] in WSNs, aiming at reducing energy consumption while not compromising end-to-end delivery delay. The assumption is that the traffic load in a WSN is usually light. However, when an event is detected, a large volume of data may need to be delivered within a short period of time. Asynchronous approaches typically seek ways to shorten preamble length so that more rooms can be left for exchanging data. For instance, X-MAC [11] uses a *strobed* preamble that consists of a sequence of short preambles prior to DATA transmission. The intervals between short preambles allow the intended receiver to send an *early ACK* to stop further preamble transmission and start DATA transmission immediately. WiseMAC [19] truncates preamble length by learning wakeup schedules of its direct neighbors, and transmits preamble just before the wakeup time of the intended receiver. RI-MAC [10], in contrast, relies on the receiver's beacon to start data transmission. Hence it does not occupy the channel until the receiver is ready for receiving. However, the broadcast of a beacon may cause collisions among multiple senders to a common receiver. It thus tries to mitigate collisions by increasing backoff window size, which is included in the receiver's beacon. Different from their contention-based method, we take advantage of the tree structure to schedule children's transmissions and thus ensure that there is no collision among contending senders. Although RI-MAC and RC-MAC both shift responsibilities to the receiver side, they target at different problems. RI-MAC tries to minimize the time that a sender occupies the channel to

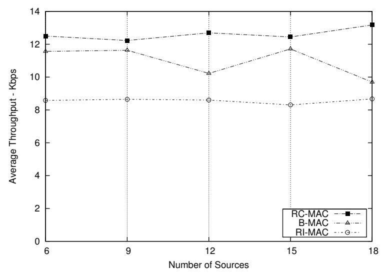

Fig. 12: Data throughput comparison in the two-hop ns-2 topology

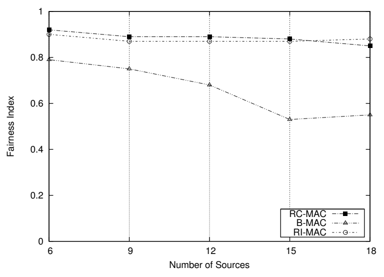

Fig. 13: Fairness index comparison in the two-hop ns-2 topology

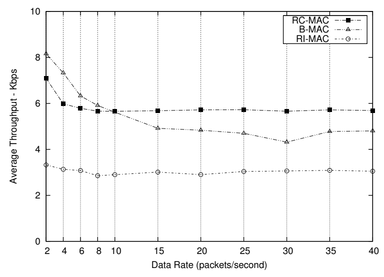

Fig. 14: Data throughput comparison in the multi-hop ns-2 topology

negotiate a time for data transmission while our RC-MAC aims to coordinate channel allocation between multiple senders.

To reduce channel collision, a group of TDMA-based protocols have been proposed [12] [20]. However, TDMA protocols may either require a global view of the entire network topology, or incur massive message exchanges between neighboring nodes. Moreover, TDMA is not flexible and scalable, which makes it impractical to be deployed in a constantly changing sensor network. Some hybrid solutions of CSMA and TDMA have been proposed recently [1] [21]. Z-MAC [1] runs a global time slots assignment at the setup phase but also allows nodes to compete for the slot if the owner does not need it. Z-MAC may degrade to CSMA when many neighboring nodes have no data to send, whereas in RC-MAC all time slots are assigned to nodes with pending data. Z-MAC improves schedule-based TDMA by introducing contention while RC-MAC enhances contention-based CSMA by incorporating scheduling. Funneling-MAC [21] adopts TDMA in the region that is close to the sink while using CSMA for the rest area. This approach can only mitigate channel contention within a small specific area by taking advantage of the sink's super ability. In contrast, RC-MAC utilizes the tree structure to assist channel access scheduling in the entire network.

VII. Conclusion

In this paper, we aim to improve data transmission throughput in case of heavy traffic loads in sensor networks through two approaches. First, we incorporate channel access scheduling into CSMA, which reduces channel collision and thus improves network throughput. Our solution is flexible and scalable, does not require any synchronization or topology information. The protocol is traffic adaptive and can be easily deployed in sensor networks. Second, we propose a sequence-based retransmission scheme in a hop-by-hop recovery pattern, which further improves communication throughput by reducing control overhead. We evaluated RC-MAC through measurements of an implementation in TinyOS on TelosB motes and detailed ns-2 simulations. Results show that the proposed RC-MAC can significantly improve data transmission throughput and achieve good fairness by utilizing the special tree structure in sensor networks.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CCF-0514078, CNS-0551464, and CNS-0721441.

References

[1] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: A hybrid MAC for wireless sensor networks," in *Proc. of SenSys*, 2005, pp. 90–101.
[2] M. Welsh, G. W. Allen, K. Lorincz, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Sensor networks for high-resolution monitoring of volcanic activity," in *Proc. of SOSP*, 2005, pp. 1–13.
[3] G. Wittenburg, K. Terfloth, F. L. Villafuerte, T. Naumowicz, H. Ritter, and J. Schiller, "Fence monitoring - experimental evaluation of a use case for wireless sensor networks," in *Proc. of EWSN*, 2007, pp. 163–178.
[4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. of SenSys*, 2009, pp. 1–14.
[5] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A highthroughput path metric for multihop wireless routing," in *Proc. of MobiCom*, 2003, pp. 134–146.
[6] S. Biswas and R. Morris, "ExOR: Opportunistic multihop routing for wireless networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 133–144, 2005.
[7] M. H. Alizai, O. Landsiedel, J. A. B. Link, S. Gotz, and K. Wehrle, "Bursty traffic over bursty links," in *Proc. of SenSys*, 2009, pp. 71–84.
[8] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. of IEEE INFOCOM*, vol. 3, Jun. 2002, pp. 1567–1576.
[9] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. of SenSys*, 2004, pp. 95–107.
[10] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proc. of SenSys*, 2008, pp. 1–14.
[11] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. of SenSys*, 2006, pp. 307–320.
[12] W.-Z. Song, R. Huang, and B. Shirazi, "TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks," in *Proc. of PerCom*, 2009, pp. 1–10.
[13] K. Klues, G. Hackmann, O. Chipara, and C. Lu, "A component-based architecture for power-efficient media access control in wireless sensor networks," in *Proc. of SenSys*, 2007, pp. 59–72.
[14] "UPMA package: Unified power management architecture for wireless sensor networks," http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/wustl/upma/.
[15] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 2000.
[16] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Digital Equipment Corp, Tech. Rep. DEC-TR-301, 1984.
[17] "CC2420 datasheet," http://www.ti.com.
[18] G. Anastasi, M. Conti, A. Falchi, E. Gregori, and A. Passarella, "Performance measurements of mote sensor networks," in *Proc. of ACM MSWiM*, vol. 1, Oct. 2004, pp. 174–181.
[19] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks," in *Proc. of ALGOSENSORS*, 2004, pp. 18–31.
[20] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *Proc. of SenSys*, 2003, pp. 181–192.
[21] G.-S. Ahn, S. G. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo, "Funneling-MAC: A localized, sink-oriented MAC for boosting fidelity in sensor networks," in *Proc. of SenSys*, 2006, pp. 293–306.