App-MAC: An Application-Aware Event-Oriented MAC Protocol for Multimodality Wireless Sensor Networks

Junzhao Du and Weisong Shi
Wayne State University
dujunzhao@gmail.com weisong@wayne.edu

***Abstract*— Urban target recognition at intersection using multimodality wireless sensor networks is very promising in reducing accidents by detecting unusual events in real time. To provide alarm signal about the incoming car to the pedestrian using sound, or notify the car driver about the pedestrian using infrastructure-to-vehicle communication, the deployed sensor system collects the sense data from multimodality sensor nodes, performs data fusion, and conducts reactions to avoid imminent accident. To address the problem, we design and implement App-MAC to support prioritized event delivery, provide inter-event and intra-event fairness, improve the performance of channel utilization, and reduce energy consumption. App-MAC leverages the advantages of contention-based and reservation-based MAC protocols to coordinate the channel access, and propose channel contention and reservation algorithms to adaptively allocate the time slots according to application requirements and current events status. To evaluate App-MAC, we have conducted simulations through TOSSIM simulator and empirical studies using Berkeley TelosB motes with target recognition events, and compared with three state-of-the-art MAC protocols, i.e., S-MAC, TDMA, and TRAMA, in terms of the proposed performance metrics, namely *average event delivery latency*, *event and sensor fairness index*, *channel utilization efficiency*, and *energy consumption efficiency*. We found that App-MAC outperforms other protocols tremendously in this application scenario.**

I. Introduction

According to the report from Intelligent Vehicle Initiative on September 2005 [1], the death resulting from intersection accidents is the deadest among all type of accidents. The death percentage of road intersection will be much higher in developing countries, e.g., China and India, because of the large population and more pedestrians. U.S. Department of Transportation lists the traffic signal violation warning as the one of the high nine high priority safety applications using wireless vehicle communication techniques, such as Dedicated Short Range Communications (DSRC) [2]. To this end, we envision that urban target recognition at intersection using multimodality wireless sensor networks is very promising in reducing accidents at intersections by detecting unusual events in real time, e.g., a pedestrian is in the middle of a road where the traffic light is green for the orthogonal direction and a car is coming fast. At this exigent moment, if the pedestrian is alarmed by a voice or the car driver is notified by an infrastructure-to-vehicle communication system, the imminent accident will avoid.

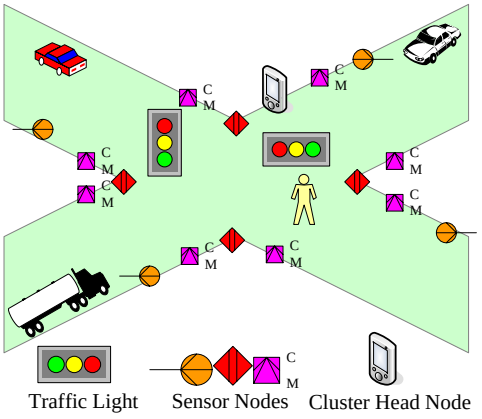

Fig. 1. The deployment example of urban target recognition system.

Fig. 1 illustrates the example deployment of wireless sensor networks for this application. In this figure, the static multimodality sensor nodes are deployed at the intersections to sense the events, e.g., the pedestrians in the middle of the road and the incoming cars to the intersection from all directions. These multimodality sensor nodes each equipped with a sensor, which includes, but is not limited to, mechanical, thermal, ultrasonic, infrared, imaging, are used to boost the recognition performance beyond the levels offered by an individual sensor acting alone. For example, the combination of ultrasonic sensor and photographic sensor can sense the distance and speed of the incoming car. The collaboration of infrared sensor and imaging sensor can sense the location of the pedestrian. Therefore, one events can be sensed by multiple sensor nodes. These sensor nodes generate variable-length event data to describe the same event from different viewpoints, e.g., image data from camera,

sound data from ultrasonic sensors, etc. Many events happen simultaneously at the intersection. Some events are urgent, e.g., a speeding incoming car and a pedestrian standing in the middle of road, while other events are ordinary. We assume the sensor node can conduct localized calculation to determine the priority of the event. A more powerful node collects the sensed events to perform data fusion. After processing the collected events, the node determines the action to perform, e.g., alarming the pedestrian or notifying the car driver. Therefore, these sensor nodes transmit the event data to the powerful node. We call this node as cluster head and other sensor nodes as cluster members. The cluster head needs to receive all of the event data from the event-correlated cluster members.

In this paper, we investigate the Medium Access Control (MAC) protocol in such a context. The event data should be transmitted with lower latency. Event delivery latency depends on the time when all the variable-length event data produced by different sensor nodes are transmitted to the cluster head. Urgent event data should be transmitted with higher priority. And the time slots also should be fairly shared with the same prioritized events (i.e., inter-event fairness). To reduce event delivery latency, time slots also should be fairly shared with the event-correlated sensor nodes (i.e., intra-event fairness). Fairness is not to allocate the time slots equally to these events and sensor nodes, but to allocate the time slots according to the real requirements of them, with large event data using more time slots. We also need to improve the performance of channel utilization to collect enough information in limited time periods. Usually, these sensor nodes are battery-powered, solar-operated or electricity-operated. Therefore, saving energy is still a big plus to extend the system lifetime.

The requirements mentioned above pose a big challenge on the design of MAC protocol. Several MAC protocols [3], [4], [5], [6], [7] have been proposed in wireless sensor networks (WSNs). And several MAC protocols [8], [9], [10], [11], [12] have been proposed specially to address these QoS issues, namely to support service differentiation. However, they have not addressed the multimodality issues in this application case. In this paper, we design and implement App-MAC in TinyOS platform [13] using Berkeley TelosB motes [14] to address the application requirements and the event-oriented and multimodality features of WSNs. App-MAC supports prioritized event delivery, fairly shares time slots with the same prioritized events (inter-event) and these event-correlated sensor nodes (intra-event), improves the performance of channel utilization, and decreases energy consumption. To evaluate App-MAC, we first propose five performance metrics for event-driven multimodality WSNs applications, then compare App-MAC with three state-of-the-art MAC protocols, namely Medium Access Control protocol with coordinated adaptive sleeping for wireless Sensor networks (S-MAC) [7], Time Division Multiple Access Control protocol (TDMA) [3], and Traffic-Adaptive Medium Access protocol (TRAMA) [5] with target recognition events. We found that App-MAC outperforms other MAC protocols tremendously in the evaluation, including decreasing average event delivery latency from 58% to 84%(simulation) and 4% to 75%(empirical study), improving the value of channel utilization efficiency from 122% to 520%(simulation) and 13% to 58%(empirical study), while at the same time improving the performance of energy consumption efficiency from 55% to 79%(simulation) and 46% to 59%(empirical study).

The contributions of the paper include three-fold:

- *Design and implement App-MAC for applications to efficiently transmit sensing data in event-oriented, multimodality WSNs.* App-MAC divides the time slots into the contention time slots (CS), the reservation time slots (RS), and the inactive time slots (IS). We also provide three general APIs, i.e., `SlotAssign`, `SlotCompeting`, and `Context`, for applications to allocate the time slots so as to provide service differentiation, and compete for the time slots so as to reduce collisions and save energy.
- *Propose RS assignment algorithm, CS assignment algorithms, and CS access protocol for optimized the allocation and contention for the time slots.* The RS assignment algorithm allocates the time slots according to application requirements and current event status. The CS assignment algorithms and CS access protocol coordinate the channel access for the spontaneous events reporting. These algorithms improve the performance channel utilization and reduce energy consumption.
- *Evaluate App-MAC using TOSSIM simulator and Berkeley TelosB motes with target recognition events and compare it with three representative MAC protocols, S-MAC, TDMA, and TRAMA.* To evaluate App-MAC, we propose five performance metrics. The *event fairness index* and *sensor fairness index* are used to evaluate the fairness of channel utilization between inter-event and intra-event. We leverage *channel utilization efficiency* to evaluate the percentage of the time slots used by successfully event transmission, while using *energy consumption efficiency* to calculate the ratio of total energy consumption to the total event data

transmission. App-MAC is compared with other three protocols in the same context using TOSSIM simulator and TelosB motes.

The rest of the paper is organized as follows. The design of App-MAC, the RS and CS assignment algorithms, and the CS access protocol are presented in Section II. Several implementation issues of App-MAC are discussed in Section III. Performance evaluation results are reported in Section IV. Related work and concluding remarks are listed in Section V and Section VI respectively.

II. App-MAC Protocol Design

In this section, we propose an application-aware, event-oriented MAC protocol (App-MAC) in multi-modality WSNs, e.g., urban target recognition at intersection. App-MAC combines the advantages of the contention-based and the reservation-based MAC protocols while offsetting their weaknesses. Furthermore, App-MAC considers more about event-oriented and multimodality features in wireless sensor networks and provides service differentiation, e.g., average event delivery latency, inter-event and intra-event fairness. App-MAC provides mechanisms to improve overall performance of channel utilization, reduce energy consumption, and support application-specific requirements in WSNs.

A. Protocol Overview

App-MAC borrows several ideas from IEEE 802.15.4 standard. To facilitate the comparison with IEEE 802.15.4 standard, we describe the basic feature of IEEE 802.15.4. A detailed description of the MAC characteristics of IEEE 802.15.4 standard is available in [15].

IEEE 802.15.4 defines a standard for a low-rate wireless personal area networks (LR-WPAN). The standard encompasses multiple frequency bands and supports various data rate. Two different device types can participate in IEEE 802.15.4 networks, i.e., a full-function device (FFD) and a reduced-function device (RFD). IEEE 802.15.4 supports two topologies, star and peer-to-peer and works in beacon-enable or non-beacon-enable mode. In the beacon-enable mode, communication is controlled by a network coordinator, which broadcasts regular beacons for synchronization and association procedures.

A superframe structure in IEEE 802.15.4 MAC is used in the beacon-enabled mode, which begins with a beacon and is followed by an active and an optional inactive period as illustrated in Fig. 2. All communication takes place in the active period, while in the inactive period, nodes can power down and save energy. The length of the superframe and the length of its active period can

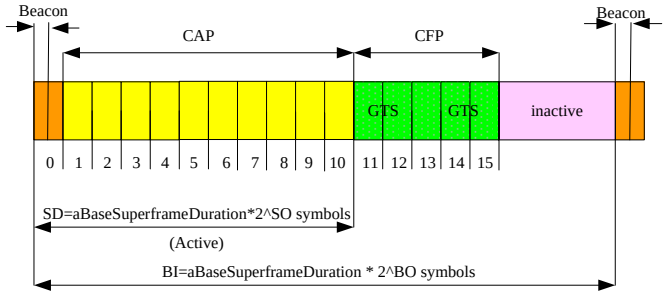

Fig. 2. The superframe in IEEE 802.15.4 (reproduced from [15]).

be tuned according to several parameters. The active period of a superframe may consist of a contention access period (CAP) and a contention free period (CFP). Channel assess in the CAP leverages a slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) [16] medium access mechanism, while the coordinator allots guaranteed time slots (GTS) in the CFP for low latency applications. In a superframe, a dedicated network coordinator, called the PAN coordinator, transmits superframe beacons in predetermined intervals. These intervals can be as short as 15ms or as long as 245s. The time between two beacons is divided into 16 equal time slots independent of the duration of the superframe. The size of the contention-free period may vary depending on demand by the associated network devices.

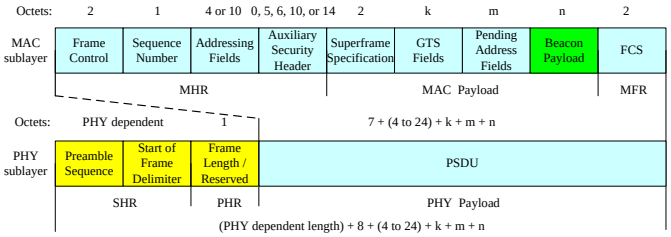

Fig. 3. The beacon frame in IEEE 802.15.4 (reproduced from [15]).

IEEE 802.15.4 MAC has four different frame types, including the beacon frame, data frame, acknowledgment frame, and MAC command frame. The beacon frame is used by a coordinator to transmit beacons. The data frame is used for all transfers of data. The acknowledgment frame is used for confirming successful frame reception. The MAC command frame is used for handling all MAC peer entity control transfers. Fig. 3 illustrates the beacon frame and the PHY packet. The detail descriptions of these frame are illustrated in [15].

Comparing with IEEE 802.15.4, App-MAC has several new features and provides more mechanisms for service differentiation, by taking event-oriented, multimodality, and application-aware feature into consid-

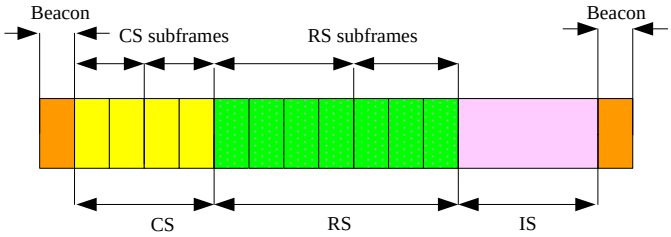

Fig. 4. The superframe in App-MAC.

eration. Similar to the superframe structure of IEEE 802.15.4 MAC, App-MAC divides the time slots in the superframe into four parts, as illustrated in Fig. 4. App-MAC assumes the time is slotted and each time slot is long enough to send one packet and tolerant of small time shift (e.g., 50ms). The first part of the superframe is the beacon time slot, which is always used by the cluster head to broadcast the beacon packet. App-MAC uses the beacon packet to synchronize the cluster members with the cluster head. Like the receiver-initiated protocols, e.g., Receiver-Initiated Busy-Tone Multiple Access protocol (RI-BTMA) [17], the cluster head polls the cluster members to transmit the event data using beacon packet. The beacon packet also acts as the ACK packets in Media Access Protocol for Wireless LAN (MACAW) [18]. The second part is the contention time slots (CS), which is used by the cluster members to report the basic event information to the cluster head when they detect new events. During these slots, App-MAC employs the CS access protocol (Section II-E), based on the slotted CSMA-CA mechanism, for channel access. The CS is further divided to several subframes. Each subframe has variable time slots. One subframe of CS is used for those sensor nodes, which have specified sensor type and detect events with priority no less than the specified one, to compete for the time slots. The following part is the reservation time slots (RS). The cluster members leverage RS to transmit the event data to the cluster head. During these slots, the cluster members access the channel exclusively without collision. The RS is also further divided to several subframes, which have variable time slots. One subframe of RS is used for a specified sensor node to transmit specified event data. The final part is the inactive time slots (IS). During these time slots, all sensor nodes just turn off the radio to reduce energy consumption. At the same time, these sensor nodes can still be on duty to detect the urgent events. In fact, the neighbor sensor can sleep in turn to save more energy and at the same time leave some active nodes to continuously sense the urgent events. Therefore, they are not supposed to keep awake in each superframe to save energy. In this case, if some urgent events happening during IS, we have still enough sensors to sense it. However, how to schedule the sensors to sleep are our next research topics.

In App-MAC, the total number of time slots in the superframe is fixed and the applications can tune it according to their requirements. For example, if we hope to increase the value of duty cycles of sensor nodes and reduce the latency of the events, we can reduce the number of time slots; if we will reduce the value of duty cycles and save more energy, we can increase the number. The number of time slots in CS, RS and IS, and the number of the subframe in CS and RS are dynamically adjusted by the cluster head through the proposed API (Section III) according to the status of events. If multiple events happen spontaneously, we increase the number of time slots of CS to reduce the collisions. If only a small number of events happen spontaneously, we reduce the time slots of CS to improve the performance of channel utilization. In the extreme case, if the number of the time slots in CS is zero, App-MAC acts as TDMA [3]. Otherwise, if the number of the time slots in RS is zero, App-MAC acts as slotted CSMA-CA [16]. Due to the fixed total time slots in one superframe, we only consider the assignment of CS and RS in this paper. Next we describe the format of the beacon packet and discuss how to assign the time slots for CS and RS in the beacon packet.

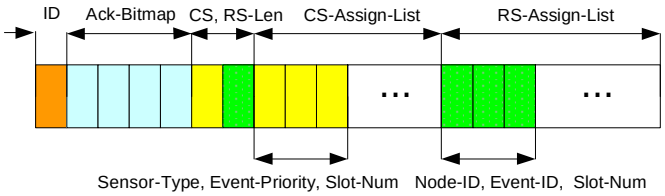

Fig. 5. The beacon packet in App-MAC.

Fig. 5 illustrates the format of the beacon packet. The cluster head uses the beacon packet to synchronize with the cluster members, acknowledge the event transmission in previous superframe, and announce the CS and RS assignments in this superframe. The first field of the beacon packet is beacon ID. Cluster members use beacon ID to identify beacon packet and prevent beacon mismatch due to the loss of the beacon packet. The second field is the acknowledgement bitmap. In the bitmap, one bit acknowledges one time slot of the previous superframe. If the cluster head has successfully received a packet in that time slot, it sets the corresponding bit as 1, otherwise 0. Using this way, the cluster head accomplishes group acknowledgements to reduce energy consumption. The following fields describe the length of

the CS and RS assignment lists in the beacon packet. Following these fields are the CS and RS assignment lists. Each assignment entry of the list describes one subframe of CS or RS. Due to the limit of the packet size, there are fixed number of assignment entries shared by CS and RS. Each CS assignment entry contains the description of this subframe, including the number of time slots (Slot-Num) in this subframe. The sensor type (Sensor-Type) and event priority (Event-Priority) specified in the assignment entry are used by App-MAC to choose sensor nodes, which are equipped with the specified sensor type and have detected events with priority no less than the specified one in the assignment entry, to compete for the time slots in this subframe. In the extreme case, if the cluster head sets the sensor type and the event priority as 0, all sensor nodes equipped with any sensor type can compete for these time slots to report any detected events. Each RS assignment entry contains a node ID (Node-ID), an event ID (Event-ID), and the number of time slots (Slot-Nun) in this subframe. The information is used by App-MAC to allocate these time slots for a specified node to transmit specified event data. The number of CS and RS assignment entries, the description in each CS and RS assignment entry are adjusted dynamically by App-MAC according to the WSNs application requirements and current events status. The application can decide whether it will change the superframe parameters dynamically or just keep them unchanged. In this paper, we design and implement the CS assignment and RS assignment algorithms for the urban target recognition application. We hope to evaluate the usability of the proposed mechanism. The details of those algorithms about how to assign CS and RS will be depicted in the following sections.

B. Functionality of Cluster Head and Cluster Members

We now describe the data exchange between the cluster head and the cluster members, and their functionality. Through the collaboration of the cluster head and the cluster members, the event data are transmitted to the cluster head in order to meet those requirements we have identified in Section I. Before the description of App-MAC protocol, we first show the data transfer model of IEEE 802.15.4 for comparison purpose.

1) Data Transfer Model in IEEE 802.15.4: In star topology of IEEE 802.15.4, two types of data transfer transactions exist. The first one is the data transfer to a coordinator. The second one is the data transfer from a coordinator. The mechanism for these transfer types also depend on whether the network supports the transmission of the beacon packets. The beacon-enable networks require synchronization with other devices. We only compare the beacon-enable model in this paper.

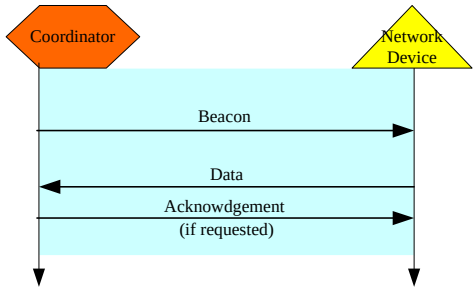

Fig. 6. Data transfer to coordinator (reproduced from [15]).

When a device wishes to transfer data to the coordinator, it first listens for the beacon packet. When the beacon packet is found, the device synchronizes to the superframe structure. At the appropriate time, the device transmits its data packet, using slotted CSMA-CA, to the coordinator. The coordinator may acknowledge the successful reception of the data by transmitting an optional acknowledgment packet. This sequence diagram is shown in Fig. 6.

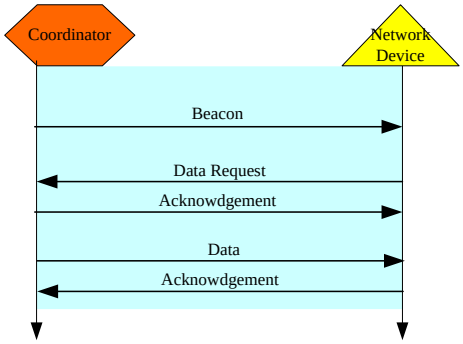

Fig. 7. Data transfer from coordinator (reproduced from [15]).

When the coordinator wishes to transfer data to a device, it indicates in the beacon packet that the data message is pending. The device periodically listens to the beacon packet and, if a message is pending, transmits a MAC command requesting the data, using slotted CSMA-CA. The coordinator acknowledges the successful reception of the data request by transmitting an acknowledgment packet. The pending data packet is then sent using slotted CSMA-CA or, if possible, immediately after the acknowledgment. The device may acknowledge the successful reception of the data by transmitting an optional acknowledgment packet. Upon successful completion of the data transaction, the message is removed from the list of pending messages in the beacon packet. This sequence diagram is shown in Fig. 7.

Next we illustrate the sequence diagram of packets exchanges in App-MAC in Fig. 8, and also describe the functionality of the cluster head and the cluster members.

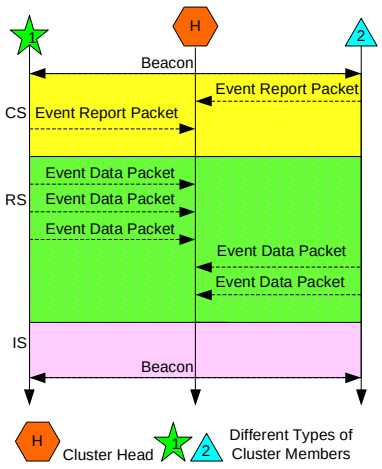

Fig. 8. The packets sequence diagram in App-MAC.

2) Functionality of Cluster Head: The cluster head performs data fusion or in-network aggregation and forwards events information to the sink node using multi-hop routing protocols. To do this, the cluster head should collect the event data from the cluster members. As shown in Fig. 8, the cluster head broadcasts the beacon packet periodically to poll the new events, acknowledges the packets transmitted by the cluster members in the previous superframe, announces the CS and RS assignments for this superframe, and synchronizes with all cluster members. The main challenging work for the cluster head is to assign the time slots in CS and RS. App-MAC has provided mechanisms to facilitate the CS and RS assignments, such as tuning the number of CS and RS assignment entries and the descriptions of these entries. But how to tune these mechanisms to meet the application requirements is not trivial. In the following section, we design and implement CS and RS assignment algorithms to address this problem. We also provide interfaces (Section III) to facilitate applications to assign CS and RS according to their special requirements.

3) Functionality of Cluster Members: The main functionality of cluster members is to detect the events, synchronize with the cluster head, and transmit the events data to the cluster head. To reduce energy consumption and improve the performance of channel utilization, the cluster members should follow the channel assignments from the cluster head. (Here we assume no malicious sensors exist.) As shown in Fig. 8, the cluster members hear the beacon packet and synchronize their time according to it. They also update their event transmitting information according to the acknowledgement in the beacon packet. When one cluster member detects an event, it should report the event to the cluster head as soon as possible. To do this, it checks the CS assignment to confirm that it meets the qualification of the CS assignment to compete for these time slots. After that, it uses the CS access protocol, (Section II-E), to compete for the time slots in CS in order to reduce collisions and save energy. When a cluster member has reported event data to the cluster head, it will check the acknowledgement in next beacon packet to make sure the cluster head has received the event reporting. If the cluster member does not get the acknowledgement of the event reporting from the cluster head, it will report the event during the next time slots in CS. Otherwise, the cluster member will not send the event reporting again and it waits for its RS assignment during the following superframes. The cluster member checks the RS assignment and transmits the specified event data in the specified time slots in RS according to the RS assignment.

C. RS Assignment Algorithm for Cluster Head

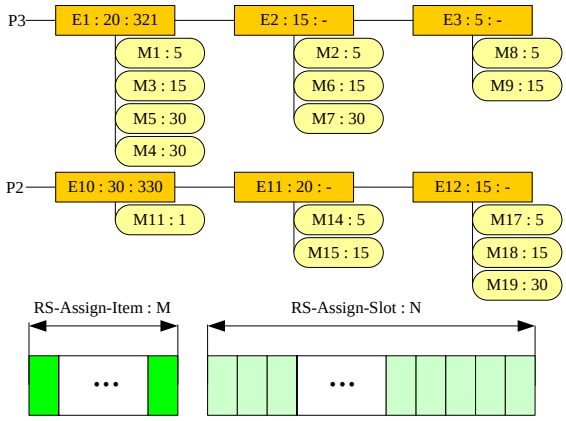

Fig. 9. An example to illustrate the RS assignment algorithm.

In IEEE 802.15.4, GTS allows a device to operate on the channel within a portion of the superframe that is dedicated (on the PAN) exclusively to that device. A single GTS may extend over one or more time slots. PAN coordinator decides whether to allocate a GTS based on the requirements of the GTS request and the current available capacity in the superframe. For each GTS, PAN coordinator shall be able to store its starting slot, length, direction, and associated device address. IEEE 802.15.4 defines the primitives for GTS allocation, deallocation, and reallocation. However, the standard only suggests that GTS shall be allocated on a first-come-first-served basis by the PAN coordinator provided there is sufficient bandwidth available. How to assign GTS to

provide the service differentiation is outside the scope of this standard. Therefore, the RS assignment algorithm complements to the IEEE 802.15.4 standard well.

The RS assignment algorithm in App-MAC is performed by the cluster head to allocate the time slots in RS to the cluster members. The objective of the RS assignment algorithm is to reduce event delivery latency, support prioritized event delivery, provide inter-event and intra-event fairness, improve the performance of channel utilization, and reduce energy consumption. Each beacon packet has several RS assignment entries and each entry assigns several time slots for a specified sensor node to transmit specified event data. The RS assignment algorithm optimizes the assignment of RS resource, including RS assignment entries and time slots in RS, in the beacon packet. We assume that there are M RS assignment entries available in this beacon packet and there are N time slots in RS (Fig. 9). The RS assignment problem now is described as how to choose (at most) M sensor nodes and assign N time slots for them to transmit their remaining packets so as to meet the objective of the RS assignment algorithm. This problem is a complex optimization problem. If we know all events information, we can calculate an optimal solution using Integer Programming [19]. However, events happen continuously and spontaneously, it is very difficult to calculate an optimal solution in this dynamic situation. Furthermore, we envision that there is no one assignment algorithm, which can get the optimal results for all applications. And finding an optimal algorithm for one particular application is doable, but beyond the scope of this paper. Here we propose an example of RS assignment algorithm to illustrate the mechanism provided by App-MAC.

To conduct RS assignment, we assume App-MAC has gotten the information of several events by previous event reporting during the time slots in CS, as shown in Fig. 9. In this Urban target recognition application, we will deploy the cluster members in such a way that when one event happens, this event can be sensed by multiple sensor nodes. These sensor nodes describe the same event from different aspects. The cluster head collects the information of the events from the cluster members. These events are stored in different queues according to their event priorities, e.g., $P3$ and $P2$ in Fig. 9. For each event, the happen time of this event is stored in the queue, then the waiting time for this event can be calculated, e.g., $E1$ has waited 20 time slots and $E10$ has waited 30 time slots. For each event, there is a queue to store the information of the sensor nodes, which have detected this event and reported the event information to the cluster head. For example, sensor nodes $M1$, $M3$, $M5$, $M4$ have detected and reported Event 1, $E1$. Hence they are in the queue of Event 1. The information of sensor nodes includes the sensor type and the remaining packets for this event to report, e.g., $M1$ has 5 packets to be transmitted.

To solve this optimization problem, we use a heuristic approach. The basic idea is that we should provide higher priority to urgent events, share the time slots with events and sensor nodes according to their requirements, and prevent starvation of events with lower priority. We propose an algorithm, weight-based event selection in multi-priority queue (WMPQ), to perform RS assignment. We also take the snapshot in Fig. 9 as an example to illustrate the WMPQ algorithm. $P2$ and $P3$ are event queues for event priority 2 and 3, respectively. Event 1, 2 and 3 are in $P3$, while Event 10, 11, 23 are in $P2$. Events in the event queue are sorted by their happen time, with the oldest event in the head. Hence Event 1 and Event 3 have the same priority and Event 1 happens before Event 3. Based on the event happen time, the cluster head can get more information about the events. If an event happens for a while, the cluster head should collect enough information. If the information is not changed for a long time, the cluster head assumes that it has collected all information about this event. However, if the event just happened, and the cluster head only receives a few number of packets, which is not enough to describe the event. In this case, the cluster head thinks that the information of this event is incomplete. Therefore, the cluster head needs wait some time slots to collect the remaining event reporting. In this snapshot, the situation is that the cluster head has collected all the information of Event 1 and only partial information of Event 3. Therefore, Event 1 has four sensor nodes, namely $M1$, $M3$, $M5$ and $M4$, and have total 80 remaining packets, while Event 3 has two sensor nodes and total 20 remaining packets. App-MAC gives higher priority to assign resource to event with higher priority. To avoid starvation of lower priority events, we use the following equation, which is inspired by our observation and initial evaluation results, to assign a weight to the event in the head of each event queue,

$$W_e = \alpha P_e + \beta T_e + \lfloor \frac{\gamma}{M_e(\lfloor \frac{K_e}{\lambda} \rfloor + 1)} \rfloor \qquad (1)$$

where P_e is the priority of the event, T_e is the waiting time of the event, defined as the time interval between the event happen time and current time. This is not the waiting time from the cluster head has received the event report to the current time. M_e is the current number of sensor nodes which have reported this event to the cluster head, and K_e is the total number of remaining packets of these sensor nodes. We describe the physical

means of the parameters as follows. α is the parameter to tuning the weight of P_e, the priority of the event. If the value of α is increased, the weight of the event with higher priority is also increased. We set α as 100 in this paper. That means the event with priority 3 has 100 credits than that with priority 2. β is the parameter to tuning the weight of T_e, the waiting time of the event. Hence, if we increase the value of β, we will assign a higher weight to event. In this paper, we choose 1 for this parameter. That means that if the node waits one time slot for get time slots in RS, it gets one credit. If the node has waited for 100 time slots, then it gets 100 credits, the same credits as event with higher priority. If we choose 2 for this parameter, the node only needs to wait 50 times slots to get 100 credits. This is an example to use aging technique to solve the starvation problem in scheduling of multiple prioritized queues. γ and λ are parameters to tuning the weight of M_e and K_e. From this equation, we can see if the event has more sensor nodes and more remaining packets to report, its weight will be smaller. We use this to give higher weight to the events with less sensor nodes and smaller number of remaining packets. λ is used to tune the weight of K_e. For example, if we set the value of as one, one remaining packet acts as one factor. If we set it as 5, 5 remaining packets act as one factor. γ is used to adjust the weight of the number of reporting sensor nodes and the remaining packets. If we increase its value, γ and λ will have more effects on the weight. In this paper, we set λ as 5, and γ as 100. That means that if we have M_e as 1 and have K_e less than 5, then it gets 100 credits, similar to a improvement of event priority. It is worth noting that this paper is a system paper and we focus on the design and implementation part of the MAC protocol. Tuning the parameters is also important for the MAC protocol design. We envision that there is no one-size-fits-all set of parameters for the optimal weight for all applications. These weights should be application-specific. In this application addressed in this paper, we choose these parameters (i.e., α=100, β=1, γ=100, λ=5), and we find that the performance is good, but we can not guarantee it is optimal. We think that finding an optimal weight for one particular application is doable, and will be our future work.

In Fig. 9, Event 1 in $P1$ happened 20 time slots ago. The cluster head found that 4 sensor nodes had reported the events and the remaining number of packets were 80. Hence, WMPQ assigns the weight of 321 to Event 1 (i.e.,P_1=3, T_1=20, M_1=4, K_1=20, and α=100, β=1, γ=100, λ=5, hence W_1=321). Similar, WMPQ assigns the weight of 330 to Event 10 in P2 (i.e.,P_{10}=2, T_{10}=30, M_{10}=1, K_{10}=1, and α=100, β=1, γ=100, λ=5, hence W_{10}=330). Event 10 has lower priority, but it has only one sensor node waiting for transmitting one packet to the cluster head. Based on this, it is better that WMPQ assigns RS resource to Event 10 to improve the overall system performance. WMPQ assigns all RS resources to one event to reduce event delivery latency. There are two cases to handle. In case one, the number of sensor nodes of this event is no larger than the number of RS assignment entries (M), and the number of total packets of this event is no larger than the time slots in RS (N). In this case, the algorithm assigns all RS resources to this event. If there are unused RS assignment entries and unused time slots in RS, the algorithm assigns them to another event. In the other case, either the number of sensor nodes or packets is more than the corresponding RS resource. To increase the value of channel utilization and to treat sensor nodes with fairness, the algorithm sorts the sensor nodes detected this event according to their remaining packets, then picks up these sensor nodes with the most number of packets, assigns all the RS assignment entries to them, and distributes all the time slots in RS to these sensor nodes according to the proportion of their remaining packets. Our evaluation results validate that this approach indeed achieves a better performance of channel utilization and provides inter-event and intra-event fairness. Note that the algorithm also adapts to the lossy links. As time goes on, it assigns more time slots for these sensor nodes with bad links.

Fig. 10 lists the pseudo-code of the RS assignment algorithm. In this algorithm, M is all sensor nodes, which have reported events to the cluster head, E is all events, P is all event queues with different priorities, mi is the number of the RS assignment entries, and ns is the number of time slots in RS. The algorithm builds multiple queues for events with different priorities, sorts each event queue according to the event happen time, sorts the sensor nodes in the event according to the number of their remaining packets, and assigns weight to each event in the queue head. After that, it chooses the event with largest weight in the head of each queue and assigns RS resource to this event. This process continues until no RS resource to use or no event to request RS resource.

D. CS Assignment Algorithms for Cluster Heads

IEEE 802.15.4 defines the primitives for communications between FFD and RFD. However, supporting service differentiation schemes in IEEE 802.15.4 is out the scope of this standard. Researchers have presented several service differentiation schemes for IEEE 802.11 [20], including scaling the contention window

The RS Assignment Algorithm

Function RS-Assign ($M[]$,$E[]$,$P[]$,mi,ns)
Begin Function

for all $m \in M \wedge e \in E \wedge p \in P$ **do**
  if $m \prec e$ **then**
    Enqueue(e,m)
  end if
  if $e \prec p$ **then**
    Enqueue(p,e)
  end if
end for
for all $p \in P$ **do**
  Sort(p)
end for
for all $e \in E$ **do**
  Sort(e)
  Weight(e)
end for
repeat
  $e \leftarrow MaxWeightInQueueHead(P)$
  $m_e \leftarrow MoteNumber(e)$
  $p_e \leftarrow PacketNumber(e)$
  if $p_e \leq ns$ **then**
    AssignAll(e)
    $E \leftarrow E - e$
  else
    AssignProportion(e)
  end if
  $mi \leftarrow mi - m_e$
  $ns \leftarrow ns - p_e$
until $mi = 0 \vee ns = 0 \vee Empty(E)$

End Function

Fig. 10. The pseudo-code of the RS assignment algorithm.

according to the priority of each flow or user, assigning different inter-frame spacing to different users, and using different maximum frame lengths for different users. In App-MAC, CS assignment algorithms can provide service differentiation for cluster members.

The CS assignment algorithms are also executed by the cluster head to assign the time slots in CS to the cluster members so as to reduce collisions among their event reporting, improve the performance of channel utilization, and reduce energy consumption. In the algorithms, the cluster head can adjust the number of the CS subframes and the description of each subframe for next superframe using the beacon packet. The description of each subframe includes the number of the time slots in this subframe and the qualification for cluster members to compete for these time slots to report their events. The qualification is that these cluster members should be equipped with specified type of sensor and have detected event with specified priority. Using this approach, the cluster head can dynamically allocate the CS resource. When the frequency of events is high, the number of the CS subframes and the number of time slots in each CS subframe should be enlarged. Applications can also use the event priority value and the sensor type value to filter out event reporting to reduce collisions. When the frequency of events is low, maybe one subframe in CS is enough and events with any priority and sensor nodes with any sensor type can compete for these time slots. Increasing the number of subframes in CS and the time slots in each subframe will reduce collisions considerably. However, it will decrease the performance of channel utilization and increase energy consumption. Therefore, it is a trade-off. Applications using App-MAC should tune the parameters according to their requirements.

Theoretically, algorithms that assign multiple CS subframes in one superframe are very complex and we intend to investigate them in the future work. In this paper, we only consider the special case that there is only one subframe in CS. We propose two algorithms to assign the CS description within one CS subframe. The first one is simple and uses a fix length of time slots in CS. The second one is more advanced and can adjust more parameters. As we know in system design, the important principle is to keep it simple and stupid (KISS), in this paper, we only implement the first algorithm. For the second one, maybe it will improve the performance. If we consider the margining improvement and system complex, we hope to choose the loss of the improvement and keep the system simple.

In first algorithm, App-MAC fixes the number of time slots in the subframe and permits any sensor type to compete for the time slots. App-MAC only adaptively adjusts the event priority qualified for the usage of this subframe. The basic idea of this algorithm is described as follows. When the cluster head receives several events reporting, it records the priority of these events. In the next beacon packet, the cluster head sets the event priority with the highest priority of the reporting events. Using this way, App-MAC filters out lower priority events reporting. In the following beacon packet, the cluster head sets the event priority with highest event priority except the previous one to permit other sensor nodes to report their events. This mechanism is very simple, but it can reduce many collisions in our evaluation.

Next we present the second CS assignment algorithm. In this algorithm, App-MAC adaptively adjusts the event priority, the sensor type and the number of time slots within one CS subframe. The basic idea of this algorithm is as follows. The algorithm initializes the priority of events to the lowest one, the type of sensor nodes as any type, and the number of time slots to the minimized one, s_{min}. In the following CS, the cluster head receives several events reporting. Due to the collisions and the

lossy links, several packets are corrupted or lost. We define the received packets as s_{pkt} and the corrupted packets as s_{bad}. If the cluster head does not receive useful packets to identify the highest priority of events, it specifies the type of the sensor node as the one produced the largest event data, and enlarges the number of time slots in the CS subframe. The next number of time slots, s_{next}, is a function of s_{min} and s_{bad}. $s_{next} = s_{min} + \alpha s_{bad}$, where α is a sensitive factor of the collisions. If the cluster head has received useful packets, it sets the priority as the highest one. This time, the algorithm has two approaches to assign the CS. In approach one, it can specify one type of sensor and set a small number of time slots. In approach two, it can specify any type of sensors and set a larger number of time slots. Both approaches have their advantages and disadvantages. The first one has higher channel utilization in CS, but may lead to lower performance of channel utilization in RS due to less event information available for next superframe. While the latter one has lower performance of channel utilization in CS, but has higher one in RS. Therefore, the algorithm adaptively chooses these approaches. When the RS channel has heavy workload, it chooses approach one, or it chooses approach two. Furthermore, in approach one, the algorithm sets s_{next} to contain all the events with specified priority. In approach two, we define s_{next} as $s_{next} = \theta n_{type}$, where θ is the event estimation parameter, n_{type} is the number of the types of sensors, which is defined by the applications. After the cluster head has received the event reporting, the parameter of the CS assignment algorithm is reset to the initialized value and this process repeats again. Note that other CS algorithms can be easily integrated with App-MAC by using the proposed APIs (Section III).

E. CS Access Protocol for Cluster Members

CSMA-CA MAC protocol is widely used in wireless networks. In IEEE 802.15.4, slotted CSMA-CA protocol with backoff is employed for CAP. App-MAC employs a modified version of CSMA-CA to support service differentiation. To make CS access protocol robust, we also need to solve the false acknowledgement problem, identified during the implementation and evaluation of App-MAC.

1) CAP access in IEEE 802.15.4: All nodes in IEEE 802.15.4 are synchronized using the beacon packet. Transmissions can begin only at the boundaries of backoff slots. Each backoff slot lasts 20 symbol durations (or 320 μs). A node which has a packet ready for transmission first backs off for a random number of backoff slots, chosen uniformly between 0 and $2^{BE}-1$, before sensing the channel. The parameter BE is the backoff exponent, which is initially set to 3. This random backoff serves to reduce the probability of collisions among contending nodes. The channel sensing mechanism then ensures that the channel is clear of activity for a contention window (CW) duration, expressed in terms of number of backoff slots. IEEE 802.15.4 standard defines the CW duration to be of 2 backoff slots (or 640 μs). If the channel is found to be busy, the backoff exponent is incremented by one and a new number of backoff slots is drawn for the node to wait, until the channel can be sensed again. This process is repeated until either BE equals the parameter $aMaxBE$ (which has a default value of 5), at which point it is frozen at $aMaxBE$, or, until a certain maximum number of permitted random backoff stages is reached, at which point an access failure is declared to the upper layer. The maximum number of permitted random backoff stages is determined by the parameter $macMaxCSMABackoffs$, which has a default value of 5.

2) CS access in App-MAC: The CS access protocol is employed by the cluster members to compete for the time slots in CS so as to report the event quickly and energy efficiently. When cluster members have detected events, they should report the event information, e.g., the event priority and the event data length, to the cluster head. Like receiver-initiated protocols, e.g., RI-BTMA [17], the cluster head in App-MAC polls the cluster members using the beacon packet to invite cluster members to report the event information. Unlike RI-BTMA, which polls one specified cluster member, App-MAC polls a couple of cluster members which meet the qualification for competing for the time slots in CS. As we have known that there are several fields in the beacon packet to describe each CS subframe, including the number time slots in the subframe and the event priority and the sensor type, which are qualified to use this subframe. Using this approach, the cluster head only permit several cluster members to compete for the time slots in each subframe. All cluster members turn on the radio to receive the beacon packet. Then the cluster members first check if they meet the qualification for one of the CS subframes, i.e., the event priority is no lower than the specified event priority and the sensor type of this node is the same as the one in the beacon packet. If the cluster member does not match any of the CS subframe, it will turn off the radio for the whole time slots of the CS. Otherwise, several cluster members meet the qualification for one of the CS subframes and they can compete for the time slots in this CS subframe. To reduce collisions caused by spontaneously transmitting, several collision avoidance mechanisms are employed in

App-MAC. In App-MAC, the cluster members randomly choose the backoff time slots before sending the packet. The backoff slots are within the time slots in this CS subframe. The number of backoff slots is related with event priority and the remaining sensor data. The cluster members, which have higher priority event or have large event data, have smaller number of backoff time slots. When a cluster member has chosen its backoff time slots, it will sense the channel of these time slots before the chosen time slot. If the cluster member overhears an event reporting before its sending, which has higher event priority, the cluster member cancels its sending to reduce the collisions with higher priority event reporting. When a cluster member cancels the event sending, it should compete for the CS slots in next superframe. And the previous action has not effects on the future action. If the channel are idle in these overhearing slots, the cluster member transmits its packet immediately after the backoff slots.

3) False Acknowledgement Problem: During the course of implementation and evaluation of App-MAC, we identify the *false ACK problem*, which is resulted from several factors, such as the lossy links and the hidden terminal problem. Here is an example of the false ACK problem. Suppose `Node A` and `Node B` both have an event to report to the cluster head. However, `Node A` and `Node B` can not sense its counterpart's usage of channel. Hence they may both choose the same time slot to report its event. Due to lossy links, the packet from `Node A` is lost, and the packet from `Node B` is received by the cluster head. In the next beacon packet, the cluster head acknowledges the received packet for that time slot. Due to the limited packet length, App-MAC does not contain the specified node ID in the acknowledgement bitmap. In this case both `Node A` and `Node B` think that the cluster head has received their reporting packet and they stop reporting their event in the following CS and begin to wait for the RS assignment. But in fact, the cluster head only receives the event reporting from `Node B` and it will not assign the time slots in RS to `Node A` in the following superframe. Therefore, `Node A` can never get RS assignment from the cluster head.

The above problem can be solved if we assume that each cluster member need several packets to report an event. With this assumption, a cluster member should be allocated several time slots in RS. If the cluster member only has one packet to send to the cluster head in the time slots in CS, we need use additional approaches to solve this, such as using additional fields in the beacon packet to direct the packet to one specified cluster member. In our implementation, we solve the problem as follows. `Node A` tries to report the event again when it finds that several later events with lower priority have received their RS assignments or there are unused RS assignment entries and unused time slots in RS of the received beacon packet. Using these methods, the node can find another chance to report the event. `Node A` only checks the beacon packet to get the information without overhearing the packet sending in time slots in RS, without generating too much overhead.

III. Protocol Implementation

We have implemented App-MAC using nesC [13] in TinyOS platform [13]. The implementation is based on Berkeley MAC protocol (B-MAC) [4], which provides bidirectional interfaces to implement other MAC protocols. The App-MAC provides three interfaces for applications to assign CS and RS according to the application requirements.

```
interface SlotAssign {
                        event  result_t  CS_Assign(Beacon_t * bc);
                        event  result_t  RS_Assign(Beacon_t * bc);
                        }

interface SlotCompeting {
                        event  result_t  CS_Competing
                             (uint8_t cur_slot, uint8_t slot_num,
                             uint8_t event_pri, uint8_t mote_type);
                        }

interface Context {
                       command  Stored_Event_t *      get_Stored_Event();
                       command Stored_Slot_t *        get_Stored_Slot();
                       command Competing_Mote_t *     get_Competing_Mote();
                       command uint32_t               get_currentTime();
                      }
```

Fig. 11. The interfaces provided by App-MAC for applications.

Fig. 11 lists the interfaces provided by App-MAC to facilitate applications. The `SlotAssign` interface is used by the cluster head to assign CS and RS resource for the next superframe using the beacon packet. Whenever the cluster head sends the beacon packet, App-MAC signals this event to applications, which can assign CS and RS time slots based on their channel utilization policies and the events status. The RS and CS assignment algorithms discussed in this paper are implemented using this interface. Other time slots assignment algorithms can be easily implemented using this interface. The `SlotCompeting` interface is used by cluster members to compete for the time slots CS during current superframe. In the beginning of each CS subframe, App-MAC signals this event to applications to inform the current information for this CS subframe, such as how many time slots are there in this subframe and what is the sensor type and event priority qualified for these time slots. The CS access protocol is implemented using this interface. App-MAC also provides the `Context`

interface for applications to get useful information, including current reported events, channel utilization, and competing sensor nodes, and so on.

Next we shortly discuss the problems we meet when we are implementing App-MAC. Due to the lossy links, several packets are lost. If the event report packets or the event data packets are lost, the cluster members can not receive the acknowledgement from the cluster head. Cluster members will retransmit the lost packets to solve this problem. However, if the beacon packet is lost, cluster members can not know the acknowledgement, RS and CS time slots assignment for this superframe. When cluster members transmit packets in this case, it leads to many collisions. Therefore, the cluster members just keep silence and wait for next beacon packet. Beacon lost sometimes leads to beacon mismatch for cluster members. Cluster members use beacon ID to identify the received beacon packet and solve the beacon mismatch problem.

IV. Performance Evaluation

We are now in the position to evaluate our proposed MAC protocol. First, five performance metrics specially for event-oriented multimodality WSNs applications are proposed. These performance metrics are used to evaluate the performance of the MAC protocols from various angles, e.g., the latency of event delivery, the fairness between events and nodes, the channel usage of clusters, and the energy consumption of the system. Second, an intensive performance evaluation of App-MAC is conducted through both TOSSIM simulator [21] and empirical studies on eight Berkeley TelosB motes [14] with target recognition events. TOSSIM can simulate the bit level transmission of packets. The advantage of using TOSSIM also includes that the nesC codes for the implementation of MAC protocols can be programmed in the real motes with little modification, the evaluation parameters can be easily configured, and the collection of the performance data is feasible. The empirical studies are to validate our simulation results and prove the real-world effectiveness of App-MAC. We also conduct a comprehensive comparison with three representative MAC protocols, i.e., S-MAC [7], TDMA [3] and TRAMA [5] in the same context. S-MAC is a contention-based MAC protocol designed for WSNs, which aims to energy conservation and self-configuration. S-MAC leverages several advanced techniques to achieve this goal, including virtual clusters, locally managed synchronization, periodic sleep and listen schedules, and collision avoidance and framing for large packets, etc. TDMA is a reservation-based MAC protocol, which is the basic line of the performance. In fact, TRAMA is hybrid of contention-based and reservation-based MAC protocols. In TRAMA, time slots are divided into random-access and scheduled-access periods. TRAMA uses a distributed election algorithm to select one transmitter within two-hop neighborhood. App-MAC borrows several idea from IEEE 802.15.4 and provides more mechanisms for service differentiation. Therefore, it is unfair for IEEE 802.15.4 to directly be compared with App-MAC in the application scenarios. In fact, App-MAC is complementary to IEEE 802.15.4 and integrating our CS assignment algorithms and RS assignment algorithm into IEEE 802.15.4 to evaluate its performance enhancement is our future work.

A. Performance Metrics

Given the inherent features of WSNs, e.g., multimodality, event-oriented, and prioritized event delivery, we propose five performance metrics in this paper to evaluate the MAC protocols in the urban target recognition application.

Event Delivery Latency. When an event happens, several nearby multimodality sensor nodes detect it and produce variable-length event data to describe it from different viewpoints. Due to the variable event data and the loss links, especially the packet schedule algorithms, it takes different time for these sensor nodes to transmit their event data to the cluster head. We define *event delivery latency* as the time period from the time the event happens to the time when all these sensor nodes finish transmitting the event data to the cluster head. In our evaluation, we record the event delivery latency for each event. Based on these data, we calculate *the average event delivery latency for all events* and *the average event delivery latency for events with specified priority*. Several factors in WSNs affect the performance of event delivery latency, such as channel schedule algorithm, event frequency, link quality and so on.

Event Fairness Index. To evaluate whether the MAC protocol treats the same prioritized events with fairness, we propose *the event fairness index*, which indicates *inter-event* fairness. The traditional definition of fairness focuses on fairly share the bandwidth to several nodes, while we define the event fairness index based on the event delivery latency of these events. The event fairness index is defined as follows.

$$I_e = \frac{1}{n}\sum_{p=1}^{n}\left(\frac{1}{n_p}\sum_{i=1}^{n_p}\sqrt{\left(L_{p_i}-\overline{L_p}\right)^2}\right) \qquad (2)$$

where L_{p_i} is the event delivery latency of the ith event with priority of p, $\overline{L_p}$ is the average event delivery latency for those events with priority p, n_p is the total

number of events with priority of p, and n is the total number of priority defined in the system. From this definition, we know that if the MAC protocol provides good inter-event fairness, the value of the event fairness index is small.

Sensor Fairness Index. To evaluate whether the MAC protocol treats the sensor nodes, which detected the same event, with fairness, we propose *the sensor fairness index*, which shows *intra-event* fairness. Sensor fairness index does not mean allocating the time slots equally among the sensor nodes, but it prefers to allocate the time slots according to the requirements of these event-correlated sensor nodes. We define sensor fairness index based on the event data delivery latency of the event-correlated sensor nodes.

$$I_m = \frac{1}{m}\sum_{e=1}^{m}(\frac{1}{m_e}\sum_{i=1}^{m_e}\sqrt{(L_{e_i}-L_e)^2}) \qquad (3)$$

where L_{e_i} is the event data delivery latency of the ith sensor node for event e, L_e is the event delivery latency of event e, m_e is the total number of sensor nodes for event e, and m is the total number of events. From this definition, we know that if the MAC protocol can allocate the time slots according to the requirements of the sensor nodes, the value of sensor fairness index is small. Intuitively, the sensor nodes with less event data are served first, and the sensor nodes with more event data are postponed for several time slots. If this case happens, the intra-event fairness is bad and the value of sensor fairness index is large.

Channel Utilization Efficiency. Channel utilization is performance metric to measure the percentage of the time slots that are used by the sensor nodes to send packets. These packets include event data packets and other packets, such as the beacon packets in App-MAC and the RTS/CTS/ACK packets in S-MAC. MAC protocols should have larger value of channel utilization to efficiently use the shared channel. However, several nodes transmit the packets during the same time slots, which cause collisions in receiver node. Hence, these time slots are wasted even they are used by transmitting packets. Therefore, we propose *channel utilization efficiency* as the percentage of time slots that are used by the sensor nodes to successfully transmit event data packets to the cluster head. In other words, channel utilization efficiency is the ratio of the total event-related packets produced in the evaluation to the total time slots used by MAC protocols to finish sending all these packets to cluster head. If MAC protocols use the time slots efficiently, the value of channel utilization efficiency should be larger.

Energy Consumption Efficiency. Energy is a vital resource for battery-powered WSNs applications. For example, the Berkeley TelosB motes [14] uses TI MSP430 microcontroller and CC2420 radio, the energy consumption of TI MSP430 in active state is 3 mW, while that in the power off state is only 15 μW. The energy consumption of CC2420 radio to transmit at 0dBm is 35 mW and that of receiving is 38 mW. To reduce energy consumption, the MAC protocols should turn off the radio as much as possible. To evaluate the energy consumption of the MAC protocol, we propose *energy consumption efficiency* as the ratio of total energy consumption to the total packets produced in the evaluation. In other words, we use this performance metric to evaluate the average energy consumption for the transmission of one packet. If the MAC protocol is energy efficiency, we have smaller value of energy consumption efficiency. In this paper, we calculate the energy consumption of all MAC protocols using the parameters provided by Berkeley TelosB motes [14].

B. Evaluation Setup

Many factors affect the performance of the MAC protocols, including the cluster size of the deployment, the link quality among the sensor nodes, the event frequency in the sensed area, the diversity of the event data produced by the multimodality sensor node, the parameters of the MAC protocols, and so on. We have conducted a comprehensive evaluation of these factors using both simulation through the TOSSIM simulator and prototype implementation based on Berkeley TelosB motes. Below we describe the setup of TOSSIM simulator and TelosB motes separately.

Simulation setup. We use simulation to evaluate App-MAC in large cluster sizes and to facilitate controlling the parameters among motes. We form a cluster with 30 motes, a reasonable cluster size in a real deployment, using TOSSIM simulator. We run TOSSIM simulator with lossy model and the packet reception rate (i.e., the link quality) among motes about 90%. In each evaluation case, there are 30 events with three event priorities. In each priority, there are ten events. These events are fired from 100 time slots to 12000 time slots (randomly) to emulate different event frequencies, from very dense events case (100 time slots) to very sparse cases. These events happen randomly at the intersection. When an event happens, three or four related motes detect it and produce their event data. We have three sensor types in the evaluation and these motes include at least one mote of these three sensor types. To be fair for other MAC protocols, we evaluate the diversity of the event data with

two group of event data: `diverse` and `similar`. For `diverse` event data, different types of motes produce 5, 10, and 30 packets for each event, respectively. While for `similar` event data, they produce 9, 10, and 11 packets for the same event. Usually, we evaluate the performance of the MAC protocols in multimodality WSNs with `diverse` event data. To testify that App-MAC also has good performance compared with other protocols in homogenous WSNs, we evaluate them with `similar` event data.

TelosB motes setup. We use empirical studies to verify the simulation results. We form a cluster using eight Berkeley TelosB motes to evaluate App-MAC. These motes are deployed on the tables around the MIST lab in Wayne State University to emulate the intersection. The packet reception rate among the motes in empirical experiments is about 90%. In each evaluation case, there are three event priorities and five events for each event priority. These events are fired within 500, 1,000, 2,000, 3,000, and 4,000 time slots (randomly). These events happen randomly at the intersection and three or four related motes detect one of the events and produce their event data. We evaluate the diversity of the event data with `diverse` and `similar` event data, which is the same as the simulation case.

For App-MAC, we set the superframe with 30 time slots, which is a reason size for the application scenario. For the RS assignment algorithm, we evaluate the parameters for the weight calculation as 100 (α), 1 (β), 100 (γ), 5 (λ). For the CS assignment algorithm, we evaluate the first algorithm and use one CS subframe and five time slots in this subframe. For S-MAC, the frame is 30 time slots and the value of duty cycles is 33%. Note that, S-MAC, TDMA and TRAMA are implemented by ourselves in TinyOS platform based on their original ideas in order to compare them in the same context.

In the rest of this section, we present the evaluation and comparison results of simulation (sim) and empirical study (real) respectively for each performance metric. We use (sim) and (real) in the caption of each figure to distinguish these two different approaches.

C. Event Delivery Latency

In multimodality WSNs, event delivery latency is affected by many factors, e.g., packet schedule algorithms, event frequency, and link quality. We compare the average event delivery latency of four MAC protocols in simulation and empirical studies.

1) Simulation: Fig. 12, 13, and 14 report the variation of the average event delivery latency for event with priority 1, 2 and 3 with event frequency. In these

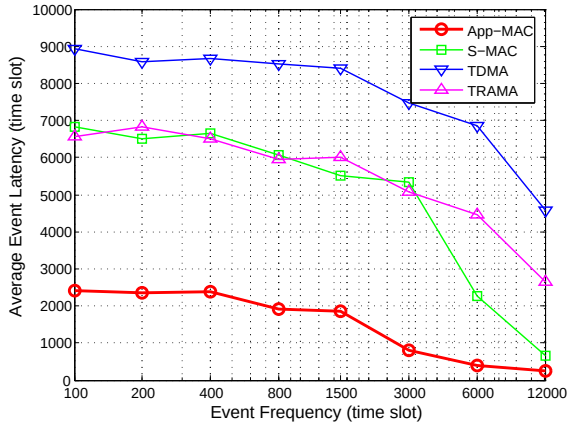

Fig. 12. The average event delivery latency of events with priority 1 vs. event frequency (sim).

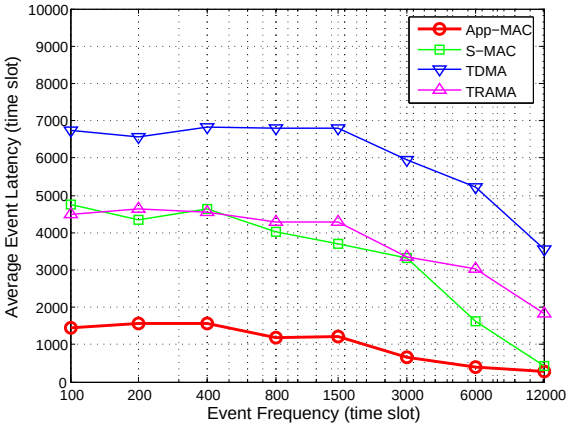

Fig. 13. The average event delivery latency of events with priority 2 vs. event frequency (sim).

figures, the x axis is event frequency, from 100 time slots to 12,000 time slots. The event frequency of 100 time slots is very dense event case, in which 300 events are fired within this time period. While the event frequency of 12,000 time slots is very sparse event case. To easily show the trends of lines, we use log scale for x axis. The y axis is average event delivery latency in term of time slots. Here one time slots is 50ms. The results are the average values of 10 runs. In each figure, we compare App-MAC, S-MAC, TDMA and TRAMA with various colors and marks for the lines.

For priority 1, the value of average event delivery latency of App-MAC ranges from 252 to 2,408 time slots; that of S-MAC ranges from 662 to 6,826; that of TDMA ranges from 4,564 to 8,922; that of TRAMA ranges from 2,639 to 6,816. The value of average event delivery latency for all MAC protocols decrease when the event dense decreases. No mater in dense event case or sparse case, App-MAC outperforms other MAC protocols, reducing the value of average event latency

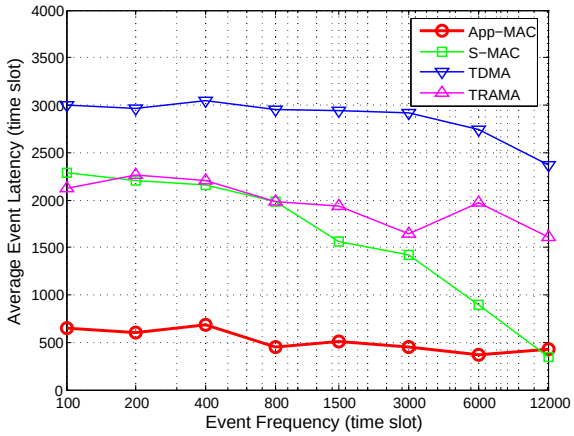

Fig. 14. The average event delivery latency of events with priority 3 vs. event frequency (sim).

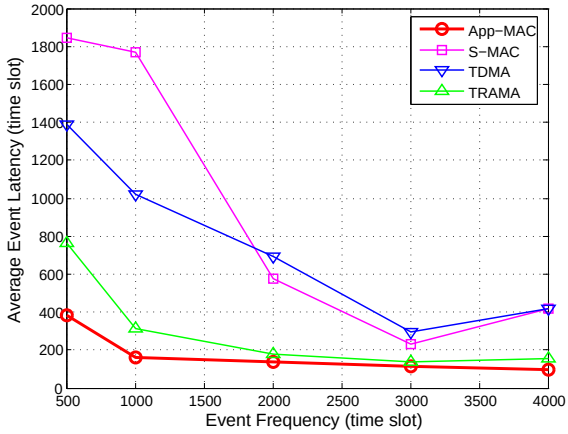

Fig. 15. The average event delivery latency of events with priority 1 vs. event frequency (real).

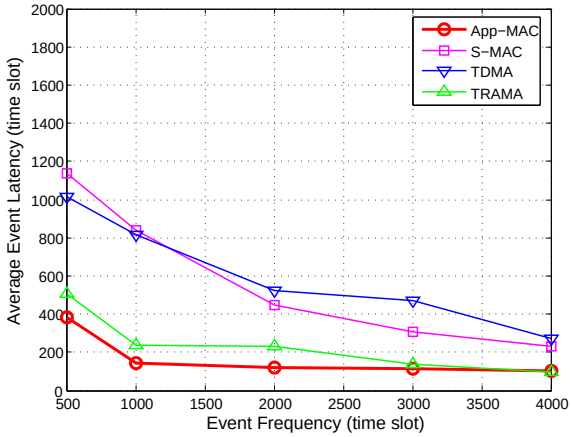

Fig. 16. The average event delivery latency of events with priority 2 vs. event frequency (real).

about 70% over S-MAC, 82% over TDMA, and 74% over TRAMA. TDMA is the worst, and S-MAC and TRAMA have the similar performance. For priority 2, the value of average event delivery latency of all MAC protocols is smaller than that of priority 1. And the performance of all MAC protocols in priority 3 is better than that in priority 2. For example, the value of the average event delivery latency of App-MAC varies from 266 to 1,576; that of others are from 420 to 4,765 (S-MAC), from 3,541 to 6,813 (TDMA), and from 1,834 to 4,634 (TRAMA) in priority 2. App-MAC outperforms about 66% (S-MAC), 84% (TDMA) and 75% (TRAMA) in priority 2. In priority 3, App-MAC reduces the value of average event delivery latency about 58% over S-MAC, 82% over TDMA, and 74% over TRAMA.

2) Empirical Study: In empirical studies, we record the value of event data delivery latency in the flash memory of the cluster head and read them out after the evaluation. Based on these raw data, we calculate the average event delivery latency for all MAC protocols.

Fig. 15, 16, and 17 compare the variation of the value of average event delivery latency for event priority 1, 2 and 3 with event frequency in empirical studies. In these figures, the x axis is the event frequency, from 500 time slots to 4000 time slots. The y axis is the average event delivery latency for the MAC protocols, i.e., App-MAC, S-MAC, TDMA and TRAMA.

The results of in empirical studies match the simulation results very well. The value of average event delivery latency decreases when the event density decreases. App-MAC is the best one. For priority 1, the value of average event delivery latency of App-MAC ranges from 96 to 383 time slots; that of S-MAC ranges from 231 to 1,844; that of TDMA ranges from 297 to 1,387; that of TRAMA ranges from 136 to 760. In priority 1, App-MAC outperforms other MAC protocols, reducing the value of average event latency about 75% over S-MAC, 75% over TDMA, and 35% over TRAMA. We also calculate the improvement of App-MAC over other MAC protocols for other priorities, including 68% (S-MAC), 72% (TDMA) and 26% (TRAMA) in priority 2, and 52% over S-MAC, 64% over TDMA, and 4% over TRAMA in priority 3. We can see that S-MAC performs worse than TDMA when event density is high in priority 1. However, in other case, S-MAC performs better than TDMA. Also we can see that TRAMA has the similar performance as App-MAC for most of event frequencies in priority 3, because of their priority-oriented design.

3) Results Analysis: From these figures in simulation and empirical study, we can see that the value of average event delivery latency of all MAC protocol decreases with decreasing of the event density. This is easy to understand. When event density is high, the channel is busy all the time and events need wait more time in order

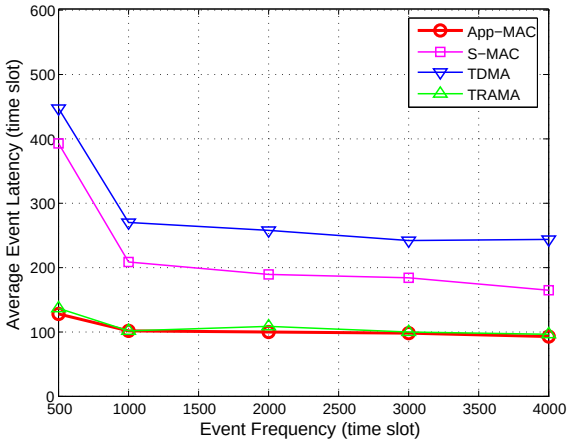

Fig. 17. The average event delivery latency of events with priority 3 vs. event frequency (real).

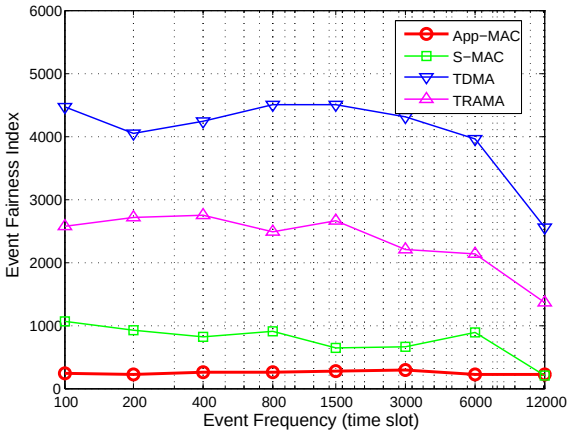

Fig. 18. The event fairness index with diverse event data vs. the event frequency (sim).

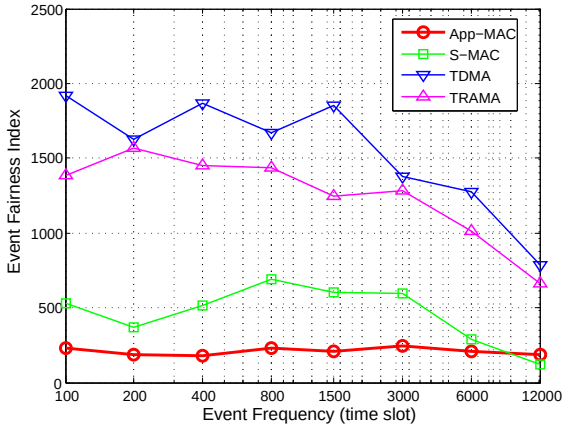

Fig. 19. The event fairness index with similar event data vs. the event frequency (sim).

to be transmitted to the cluster head. Therefore, the value of average event delivery is large in dense cases and is small in sparse cases.

We also find that all MAC protocol have provides higher priority to serve the event with higher priority. In fact, our implementation policy magnify this effects. When we implement other MAC protocols, we give higher priority to allocate the channel slots for the event with higher priority.

We attribute this significant improvement to the fact that in App-MAC the CS assignment algorithms and the CS access protocol can filter out lower prioritized events and reduce the collisions caused by spontaneous event reporting, and the RS assignment algorithm supports prioritized delivery of events, guarantees the urgent events transmitting without interrupting from others, and adaptively assigns more time slots to motes with more data.

D. Event Fairness Index

Event fairness index shows inter-event fairness. The definition of event fairness index is based on the event delivery latency of these events. If the events with the same priority have similar value of event delivery latency, the performance of event fairness index is better and the value of this performance metric is smaller. The event frequency and the diversity of the event data have significant impacts on event fairness index. To be fair to other MAC protocols, we evaluate this performance metric with both `diverse` and `similar` event data in various event frequencies. Both simulation results and the empirical studies are illustrated here.

1) Simulation: Fig. 18 reports the variation of the event fairness index for all MAC protocols with event frequency for `diverse` event data, while Fig. 19 is that using `similar` event data. In these figures, the x axis is event frequency, from 100 time slots to 12,000 time slots, and the y axis is the value of event fairness index. For `diverse` event data, the values of event fairness index of App-MAC, S-MAC, TDMA and TRAMA are ranging from 212 to 284, 203 to 1,063, 2,660 to 4,499, and 1,357 to 2,745, respectively, while those with `similar` event data are ranging from 181 to 230, 123 to 689, 789 to 1,920, and 664 to 1,570, respectively. According to the definition of event fairness index, if the value is smaller, the MAC protocol supports inter-event fairness better.

From these figures, we can see all MAC protocols perform better for `similar` event data that they do for `diverse` event data. App-MAC is the best one. S-MAC is better than TDMA and TRAMA. TDMA is the worst. For `diverse` event data, App-MAC improves the performance of event fairness index about 60% over S-MAC, 94% over TDMA, and 89% over TRAMA. For `similar` event data, App-MAC outperforms about 42%

over S-MAC, 86% over TDMA, and 82% over TRAMA.

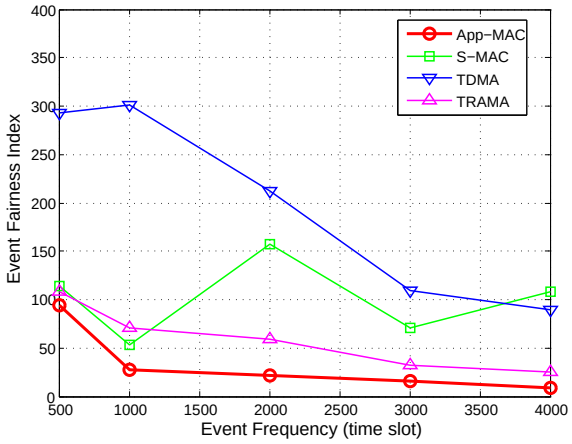

Fig. 20. The event fairness index with diverse event data vs. the event frequency (real).

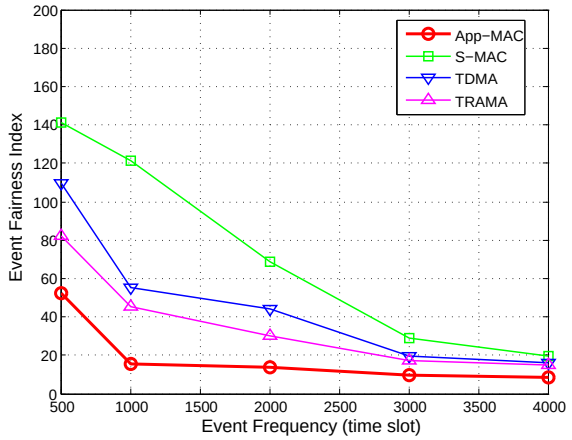

Fig. 21. The event fairness index with similar event data vs. the event frequency (real).

2) Empirical Study: Fig. 20 reports the variation of the event fairness index for all MAC protocols with event frequency using `diverse` event data, while Fig. 21 reports that using `similar` event data. In these figures, the x axis is the event frequency, from 500 time slots to 4000 time slots. The y axis shows the value of event fairness index.

For `diverse` event data, the values of event fairness index of App-MAC, S-MAC, TDMA and TRAMA are ranging from 8 to 95, 53 to 158, 90 to 301, and 26 to 108, respectively, while those with `similar` event data are ranging from 8 to 53, 20 to 141, 16 to 110, and 15 to 82, respectively.

The results of empirical study have a little difference from the results of the simulation data. S-MAC performs worse than TRAMA and better than TDMA for `diverse` event data. For `similar` event data, S-MAC is the worst. App-MAC is the best for all cases. For `diverse` event data, the performance of event fairness index for App-MAC improves about 64% over S-MAC, 85% over TDMA, and 51% over TRAMA. For `similar` event data, the improvement of the performance of App-MAC is about 71% over S-MAC, 59% over TDMA and 52% over TRAMA.

3) Results Analysis: These results illustrate that, in term of inter-event fairness, all MAC protocols perform better with `similar` event data than they do with `diverse` event data. We attribute this to the fact that the event delivery latency for `similar` event data has smaller variation than that for `diverse` event data. Hence, the value of event fairness index is smaller. S-MAC favors to transmit event data in a group. That means S-MAC only finishes transmission of one event data, it begins to transmit other data. This behavior hurts its performance of event fairness index. TRAMA allocate the time slots proportional to the event data using its distributed coordination mechanisms. Hence, TRAMA performs well in this performance metrics. No matter with `diverse` event data and `similar` event data, App-MAC outperforms other MAC protocols about 60% to 94% (diverse) and about 42% to 86% (similar) for simulation, and 51% to 85% (diverse) and 52% to 71% (similar) for empirical studies, contributed by the RS assignment algorithm in App-MAC.

E. Sensor Fairness Index

The sensor fairness index indicates intra-event fairness. Similar with event fairness index, the event frequency and the diversity of the event data also have impacts on the value of sensor fairness index. We also evaluate them with both `diverse` and `similar` event data in different event frequencies for simulation and empirical studies.

1) Simulation: Fig. 22 reports the variation of sensor fairness index for all MAC protocols with event frequency using `diverse` event data in, while Fig. 23 illustrates that using `similar` event data. In these figures, the x axis is event frequency, range from 100 to 12,000 time slots. The y axis represents sensor fairness index. With `diverse` event data, the value of sensor fairness index of App-MAC, S-MAC, TDMA, and TRAMA varies from 96 to 229, 120 to 705, 1,407 to 2,396 and 654 to 1,378, respectively. With `similar` event data, the value for App-MAC, S-MAC, TDMA, and TRAMA varies from 83 to 179, 72 to 511, 364 to 965 and 506 to 1,100, respectively. These results illustrate similar results as that of event fairness index.

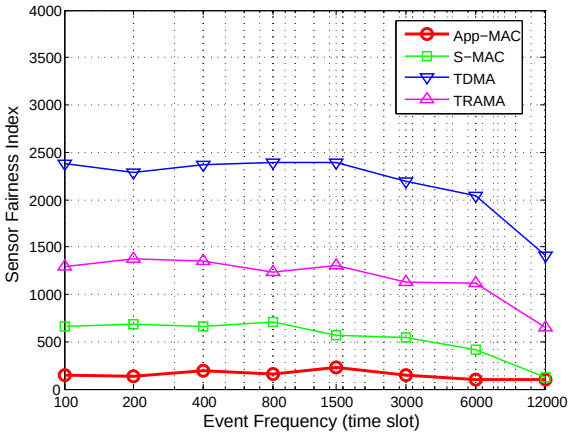

Fig. 22. The sensor fairness index with diverse event data vs. event frequency (sim).

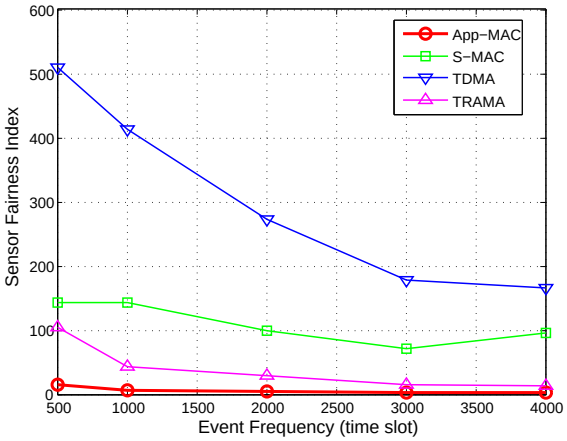

Fig. 24. The sensor fairness index with diverse event data vs. event frequency (real).

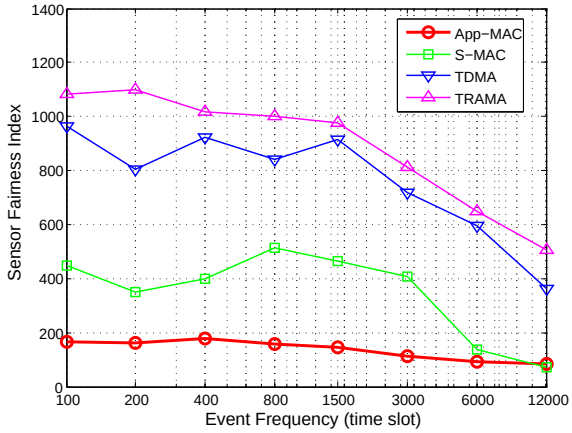

Fig. 23. The sensor fairness index with similar event data vs. event frequency (sim).

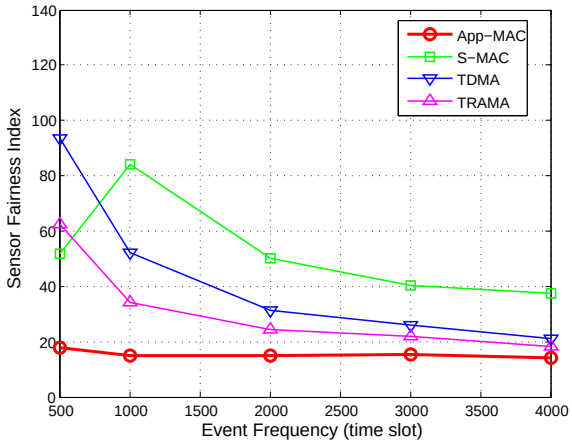

Fig. 25. The sensor fairness index with similar event data vs. event frequency (real).

The performance of all MAC protocols in `similar` event data is better than that in `diverse` event data. App-MAC is the best one. The performance of S-MAC is better than that of TDMA and TRAMA. App-MAC improves the performance of sensor fairness index about 67% over S-MAC, 93% over TDMA, and 87% over TRAMA for `diverse` event data, while about 50% over S-MAC, 82% over TDMA, and 85% over TRAMA for `similar` one.

2) Empirical Study: Fig. 24 reports the variation of sensor fairness index for all MAC protocols with event frequency using `diverse` event data, while Fig. 25 illustrates that using `similar` event data. In these figures, the x axis is event frequency, which is from 500 time slots to 4000. The y axis is sensor fairness index.

With `diverse` event data, the value of sensor fairness index of App-MAC, S-MAC, TDMA, and TRAMA varies from 2 to 16, 72 to 142, 165 to 510, and 12 to 105, respectively. App-MAC improves the performance of sensor fairness index about 95% over S-MAC, 98% over TDMA, and 83% over TRAMA. With `similar` event data, the intra-event fairness values of App-MAC is smallest, which shows that App-MAC performs well even in homogeneous WSNs. The performance of App-MAC improves about 69% over S-MAC, 26% over TDMA, and 53% over TRAMA. Again, S-MAC is better than TDMA in with `diverse` event data. While it is worst for `similar` one.

3) Results Analysis: These results illustrate similar results as those of event fairness index. No matter with `diverse` event data and `similar` event data, App-MAC outperforms other MAC protocols about 67% to 93% (diverse) and 50% to 85% (similar) in simulation. In empirical studies, the improvement of App-MAC is about 83% to 98% (diverse) and 26% to 69% (similar).

In App-MAC, the CS assignment algorithms and the CS access protocol provide the motes, which the have same prioritized events, with the same probability to

compete for the time slots in CS, and the RS assignment algorithm of App-MAC fairly allocates the time slots in RS to cluster members. These mechanisms improve event and sensor fairness index significantly.

F. Channel Utilization Efficiency

Channel utilization is the performance metric to evaluate the usage of radio channel by the MAC protocols. In TOSSIM simulator, we record the channel usage information of each time slot, which is used by the motes to send packets. Based on these data, we depict the detailed information of the accumulated channel utilization varied with the time slots. The accumulated channel utilization is the percentage of occupied channel to the total time slots till current sample time slot. After that, we compare the channel utilization efficiency, a performance metric to measure the efficient usage of channel, in simulations and empirical studies.

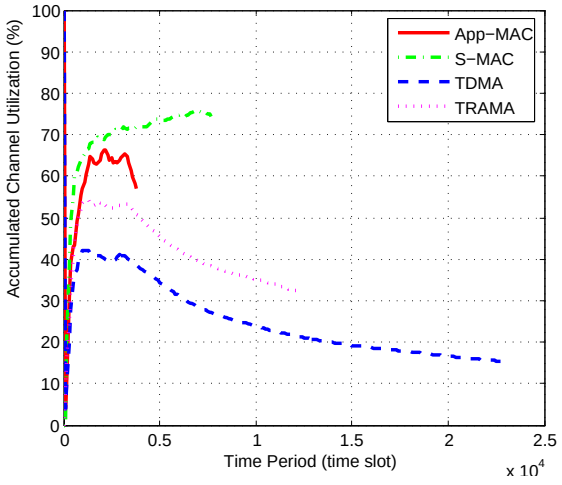

Fig. 26. The accumulated channel utilization for 3000 time slots (sim).

1) Simulation: Fig. 26 depicts the accumulated channel utilization of App-MAC, S-MAC, TDMA and TRAMA in the case that the events are fired within 3000 time slots. In this figure, the x axis is the time period and the y axis is the accumulated channel utilization. The channel utilization of App-MAC has a peak value of about 66%, and then drops down for a short time period, finally ends after 3,847 time slots. The channel utilization of S-MAC has a peak value of about 76%, and keeps from 60% to 76%, then end after 7,710 time slots. At the beginning, TDMA and TRAMA have a very larger value of channel utilization. Then their values of channel utilization drop gradually to from 43% to 15%, and 55% to 32%, respectively, and this low value state lasts a long time. TDMA ends after 22,879 time slots, while TRAMA ends after 12,106 time slots. The end of the lines means that end of the events data transmission. This figure reflects the working mechanisms of these MAC protocols. For TDMA and TRAMA, at the beginning time period, most motes have event data to transmit. So the value of channel utilization efficiency is large. With the time goes on, there are only few motes, which produce large event data for each event and detect more events, have event data to transmit. Therefore, the value of channel utilization efficiency keeps in a small amount and lasts for a long time. In S-MAC, when cluster members have event data, they compete for the channel to send RTS packets, when one cluster member wins the CTS, it sends DATA packets to the cluster head, while receiving ACK packets from it. These mechanisms make the channel busy all the time slots. Therefore, the value of channel utilization is higher. However, the control packets use more time slots, which is the overhead. Hence, the total time slots used to transmit packets is still larger than that of App-MAC. In App-MAC only a small number of motes compete for the time slots in CS using the adaptive CS assignment algorithms and the CS access protocol. Due to the RS assignment algorithm, App-MAC assigns the time slots in RS to those event-correlated motes. These mechanisms together reduce the collisions, improve the fairness of events and motes, and also reduce event delivery latency.

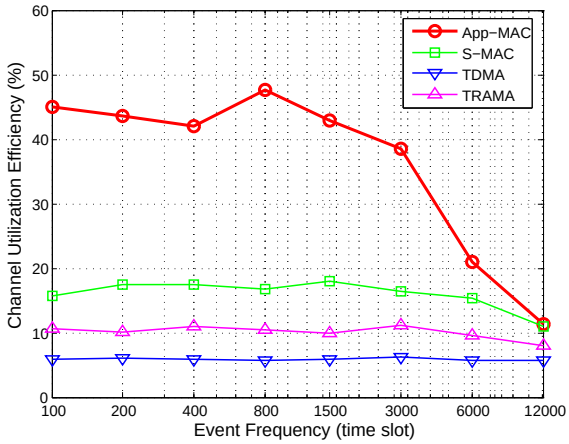

Fig. 27. Channel utilization efficiency vs. event frequency (sim).

Note that the channel utilization of App-MAC is lower than that of S-MAC, however, App-MAC uses much less time slots than that of S-MAC to finish transmitting all the event data. This testifies that App-MAC uses the channel more efficiently than other MAC protocols, as shown in Figure 27, which reports the variation of the channel utilization efficiency of App-MAC, S-MAC, TDMA and TRAMA with event frequency. In this figure, the value of channel utilization efficiency of

App-MAC, S-MAC, TDMA, and TRAMA varies from 11% to 48%, 11% to 18%, 5% to 6%, and 8% to 11%, respectively. App-MAC has large value of channel utilization efficiency in dense event cases. When the event density decreases, the performance of App-MAC drops quickly. All other MAC protocols have lower performance. S-MAC is better than TRAMA, which is better than TDMA. App-MAC improves the performance of channel utilization efficiency 122% over S-MAC, 520% over TDMA, and 255% over TRAMA. In term of channel utilization efficiency, App-MAC is much better than all these MAC protocols.

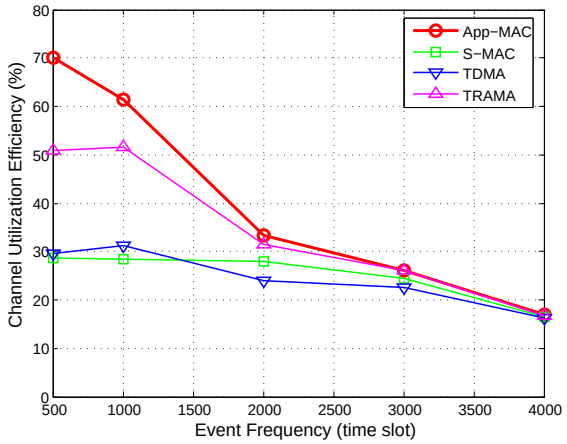

Fig. 28. Channel utilization efficiency vs. event frequency (real).

2) Empirical Study: Fig. 28 shows the variation of the channel utilization efficiency of App-MAC, S-MAC, TDMA and TRAMA with event frequency. In this figure, the value of channel utilization efficiency of App-MAC, S-MAC, TDMA, and TRAMA varies from 17% to 70%, 16% to 29%, 16% to 31%, and 17% to 52%, respectively. In this figure, the performance of all MAC protocols drops when event density decreases. The performance of App-MAC is the best one. The performance of TRAMA is better than that of S-MAC. From the figure, it can be seen that App-MAC improves the performance of channel utilization efficiency 58% over S-MAC, 59% over TDMA, and 13% over TRAMA in empirical experiments.

3) Results Analysis: From the simulation results and empirical experiments, we can see that the performance of App-MAC is much better than that of others. Again, the mechanisms in App-MAC, e.g., the CS assignment algorithm, the CS access protocol, and the RS assignment algorithm, adaptively allocate the time slots to motes so as to improve the performance of channel utilization efficiency.

G. Energy Consumption Efficiency

The energy consumption of all protocols increases as the time goes on. In the contention time slots of S-MAC, when motes have event data, all of them compete for the time slots. At most one of them gets the CTS packet and other motes try to compete for the channel again in next contention time slot. In the packet transmitting time slots, S-MAC always sends ACK to each received packet to solve the hidden terminal problem. These mechanisms consume more energy and extend the total time slots to finish transmitting all the event data. In TDMA and TRAMA, motes transmit packets to the cluster head when they have event data, otherwise they just turn off the radio. The cluster head in TDMA and TRAMA turns on the radio all the time to receive packets from cluster members. Due to their fixed channel assignment scheme, it takes a long time to finish sending all the event data. TRAMA consumes less energy than that of TDMA using the weighted channel assignment mechanism. App-MAC consumes the least energy among all the protocols. It takes less time to finish transmitting packets, because App-MAC allows only part of motes to compete for the time slots in CS using its adaptive CS assignment algorithms. Also, App-MAC lets several motes to withdraw competing for the time slots in its CS access protocol. The adaptive CS and RS assignment algorithms also reduce idle listening and turn off the radio of motes as much as possible. The following figures validate our expectation.

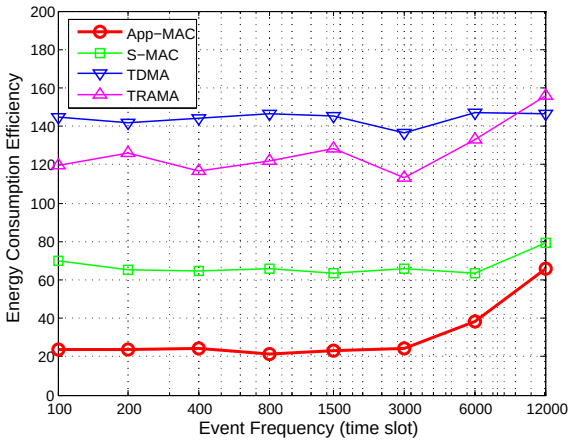

Fig. 29. Energy consumption efficiency vs. event frequency (sim).

1) Simulation: Fig. 29 reports the variation of the energy consumption efficiency for App-MAC, S-MAC, TDMA and TRAMA with event frequency. The value of energy consumption efficiency of App-MAC, S-MAC, TDMA, and TRAMA varies from 22 to 66, from 64 to 79, from 137 to 147, and from 113 to 156, respec-

tively. App-MAC improves the performance of energy consumption efficiency about 55% over S-MAC, 79% over TDMA, and 77% over TRAMA.

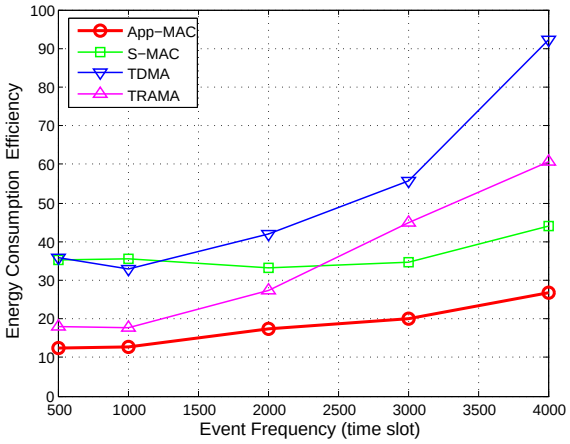

Fig. 30. Energy consumption efficiency vs. event frequency (real).

2) Empirical Study: Fig. 30 reports the variation of the energy consumption efficiency for App-MAC, S-MAC, TDMA and TRAMA with event frequency. The value of energy consumption efficiency of App-MAC, S-MAC, TDMA, and TRAMA varies from 12 to 27, from 33 to 44, from 33 to 92, and from 18 to 61, respectively. App-MAC improves the performance of energy consumption efficiency 52% over S-MAC, 64% over TDMA, and 41% over TRAMA in empirical studies.

3) Results Analysis: These results illustrate that App-MAC is more energy efficiency than other MAC protocols. We attribute this to the fact that, in App-MAC, the CS assignment algorithm and the CS access protocol constrain the motes to compete for CS so as to reduce energy consumption, and the RS assignment algorithm fairly allocates RS to reduce the total time slots to finish transmitting the event data.

To this end, we conclude that App-MAC is better than the three other protocols in term of the average event delivery latency, event and sensor fairness indexes, the channel utilization efficiency and energy consumption efficiency.

V. Related Work

MAC protocols have been extensively studied in wireless networks [22], mobile ad-hoc networks [23], and wireless sensor networks [24]. App-MAC is built upon IEEE 802.15.4, and we have discussed extensively their difference in the design part of the paper. In this section, we compare these MAC protocols, which have heavy impact on the design of App-MAC protocol.

A. Related MAC Protocols in Wireless Networks and Mobile Ad-Hoc Networks

Another alternative for computer communications (ALOHA) [25] uses wireless broadcasting to create single hop radio networks. In ALOHA, a node may access the channel as soon as the data is ready. Naturally, more than one node may transmit at the same time, thus causing collisions. Slotted-ALOHA [26] introduces synchronized transmission time slots. In this case, nodes can transmit only at the beginning of a time slot. Slotted-ALOHA doubles the throughput compared to ALOHA, with the cost of necessary time synchronization.

To improve the throughput and solve the hidden-terminal and the exposed-terminal problems, additional collision avoidance or collision detection methods have been employed. In Carrier Sense Multiple Access protocol (CSMA) [27], the transmitting node first senses the medium to check whether it is idle or busy. If the medium is busy, the node defers its own transmission to prevent a collision with the existing transmission. Otherwise, the node begins to transmit its data. Multiple Access with Collision Avoidance protocol (MACA) [28] uses the Request-To-Send/Clear-To-Send (RTS/CTS) control packets to prevent collisions. Media Access Protocol for Wireless LAN's (MACAW) [18] is proposed using RTS-CTS-DS-DATA-ACK exchange to allow much faster error recovery at the data link layer. Floor Acquisition Multiple Access protocol (FAMA-NCS) [29] is a combination of control packets and carrier sensing.

The above MAC protocols are sender-initiated protocols. Multiple Access with Collision Avoidance by Invitation (MACA-BI) [18] and Receiver-Initiated Busy-Tone Multiple Access (RI-BTMA) [17] are receiver-initiated protocols, i.e., the receiving node polls a potential transmitting node for data. If the sending node indeed has data for the receiver, it is allowed to transmit after being polled.

Time Division Multiple Access protocol (TDMA) [3] is collision-free medium access protocol. In TDMA, time period is divided into frames that provide each node with a transmission time slot over which it can transmit data without collisions. However, TDMA needs time synchronization and the throughput at low traffic loads is low due to idle time slots. Furthermore, finding an efficient time schedule in a scalable fashion and handling dynamic topology changes are also not trivial.

Several hybrid MAC protocols combine the contention-based and collision-free approaches. In Centralized Packet Reservation Multiple Access (C-PRMA) [30], nodes use the random access time slots to

send their QoS requirements to the base station (BS). The BS schedules the uplink transmissions, taking into account different traffic rates and delay constraints. Reservation Random Access with Independent Stations Algorithm (RRA-ISA) [31] uses Independent Stations Algorithm to distribute the right to transmit in a time slot among nodes so as to maximize the throughput from slot to slot. Distributed Queuing Request Update Multiple Access (DQRUMA) [32] has good performance with Guaranteed Bandwidth and Minimum Delay (GBMD) scheduling. In Mobile Access Scheme based on Contention and Reservation for ATM (MASCARA) [33], the BS collects all the requests and makes time slot assignments using the Prioritized Regulated Allocation Delay-Oriented Scheduling (PRADOS) algorithm.

To save scare energy for handhold device or embedded device, Power Aware Multiple Access protocol with Signaling (PAMAS) [34], a power aware MAC protocol, is proposed to achieve significant power savings by powering down nodes at the appropriate times. Furthermore, Power Control MAC (PCM) [35] uses power control by adaptively regulating transmission power levels according to many factors.

IEEE has standardized the IEEE 802.11 protocol for Wireless Local Area Networks [16]. IEEE 802.11 specifies two modes of MAC protocol: Distributed Coordination Function (DCF) mode (for ad hoc networks) and Point Coordination Function (PCF) mode (for centrally coordinated infrastructure-based networks). DCF is a CSMA-CA scheme with binary slotted exponential backoff, which can be seen as a combination of the CSMA and MACA schemes. In DCF, backoff and IFS (SIFS/PIFS/DIFS) are used to regulate medium access. IEEE 802.11 DCF mode has a power saving mechanism. At the beginning of each beacon interval, each node must stay awake for a fixed time called ATIM window. After that, several nodes can go into sleep state immediately.

In IEEE 802.11 protocol, packets that have missed their deadlines are still retransmitted, even though they are not useful any more. Several QoS-Aware protocols, e.g., DCF with Priority Classes (DCF-PC) [36], Elimination by sieving DCF (ES-DCF) [37], MACA with Piggyback Reservation (MACA/PR) [38], are extensions of IEEE 802.11. The basic ideas are to use a combination of shorter IFS or waiting times and shorter backoff time values for higher priority data. These schemes have been shown to achieve drastic reductions in mean packet delay, missed deadlines, and packet collisions as compared to IEEE 802.11.

Distributed Fair Scheduling (DFS) [39] ensures that different flows sharing a common wireless channel are assigned appropriate bandwidth corresponding to their weights or priorities. The fundamental idea of DFS is that each packet is associated with start and finish time stamps. A higher priority packet is assigned a smaller finish-tag and shorter backoff periods. In DFS, the start and finish times for packets are calculated on the basis of the Self-Clocked Fair Queuing (SCFQ) algorithm.

B. Related MAC Protocols in Wireless Sensor Networks

Medium Access Control with coordinated adaptive sleeping for wireless Sensor networks (S-MAC) [7] is a contention-based MAC protocol designed for WSNs, which aims to energy conservation and self-configuration. Locally managed synchronization and periodic sleep and listen schedules based on these synchronization are the basic idea of S-MAC. Neighbor nodes form virtual clusters to set up a common sleep schedule. The listen interval is divided into three parts, i.e., SYNC, RTS, and CTS. Each part is further divided into many time slots for senders to perform carrier sensing. Schedule exchanges are accomplished by periodical SYNC packet broadcasting to immediate neighbors. Collision avoidance is achieved by a carrier sensing. RTS/CTS packet exchanges are used for unicast of data packets. Long messages are divided into frames and sent in a burst. With those techniques, one may achieve energy savings by minimizing communication. Adaptive Energy-Efficient MAC (T-MAC) [6] is proposed to enhance the performance of S-MAC under variable traffic load. In T-MAC, listen period ends when no activation event has occurred for a time threshold TA.

Traffic-Adaptive Medium Access protocol (TRAMA) [5] is a TDMA-based algorithm and proposed to increase the utilization of classical TDMA in an energy efficient manner. It is similar to Node Activation Multiple Access (NAMA) [40], where for each time slot a distributed election algorithm is used to select one transmitter within two-hop neighborhood. This kind of election eliminates the hidden terminal problem and hence, ensures all nodes in the one-hop neighborhood of the transmitter will receive data without any collision. Time is divided into random-access and scheduled-access periods. Random-access period is used to establish two-hop topology information where channel access is contention-based. There are three components in TRAMA, i.e., the Neighbor Protocol (NP), the Schedule Exchange Protocol (SEP), and the Adaptive Election Algorithm (AEA). NP propagates one-hop neighbor information among neighboring nodes during the random access time slots using the signaling time slots. Transmission time slots are used

for collision-free data exchange and also for schedule propagation. A node has to announce its schedule using SEP before starting actual transmissions. AEA selects transmitters and receivers to achieve collision-free transmission using the information obtained from NP and SEP and uses traffic information to improve the performance of channel utilization efficiency.

A hybrid MAC for wireless sensor networks (Z-MAC) [41] combines the strengths of TDMA and CSMA while offsetting their weaknesses. The current implementation of Z-MAC uses Distributed Randomized dining philosophers (DRAND) [42] to assign the time slots to each node in the network. The TF rule allows nodes to pick their own time frame sizes based on their local two-hop information. In Z-MAC, a node can be in one of two modes: Low Contention Level (LCL) or High Contention Level (HCL). A node is in HCL only when it receives an Explicit Contention Notification (ECN) message from a two-hop neighbor within the last ECN period.

Berkeley MAC (B-MAC) [4] is a lightweight MAC protocol used as the default MAC for Mica2. B-MAC allows application to implement its own MAC through a well-defined interface. They also adopt Low Power Listening (LPL) and Clear Channel Sensing (CCA) technique to improve the performance of channel utilization efficiency.

A short preamble MAC protocol for duty-cycled wireless sensor networks (X-MAC) [43] proposes solutions to the problems caused by the long preamble, which introduces excess latency at each hop and suffers from excess energy consumption at nontarget receivers. X-MAC employs a shortened preamble approach, which reduces energy usage at both the transmitter and receiver, reduces per-hop latency, and adapts both bursty and periodic sensor data sources.

A localized and sink-oriented MAC for boosting fidelity in sensor Networks (Funneling-MAC) [44] exhibits a unique funneling effect in many-to-one and hop-by-hop traffic pattern in sensor networks. The funneling-MAC is based on a CSMA/CA and also implements a localized TDMA algorithm in the funneling region. The sink node manages the TDMA scheduling of sensor events in the funneling region. TDMA only operates locally in the funneling region close to the sink and not across the complete sensor field.

C. Special Features of App-MAC Protocol

To our knowledge, App-MAC is the first MAC protocol that takes multimodality feature of wireless sensor networks into consideration. App-MAC groups the sensor nodes which detect the same event from different viewpoints and allocates the time slots for these sensor nodes according to the event priority and variable-length event data. App-MAC also provides service differentiation for the events with higher priorities.

App-MAC borrows the idea of the superframe from IEEE 802.15.4 MAC, grouping the time slots in CS, RS and IS. The times slots in CS, RS and IS are dynamically adjusted according to the application requirements and event status. App-MAC combines the strengths of contention-based and reservation-based MAC protocols while offsetting their weaknesses. App-MAC uses the contention-based protocol to report event information. Several mechanisms are employed to handle collision avoidance and collision detection. Like receiver-initiated protocols, e.g., MACA-BI and RI-BTMA, the cluster head in App-MAC polls the cluster members using the beacon packet to invite cluster members to report event data. App-MAC polls several cluster members which have event data with specified event priority and specified sensor type while not polling one specified node as MACA-BI does, thus reducing the collisions from potential sending. The CS assignment algorithms run by the cluster head assign the time slots in CS to filter out simultaneous event reporting using the collected event information. On the other hand, the CS access protocol run by the cluster members is to reduce collisions and save energy. Like MACA, cluster members compete for one time slot in CS by sending the event reporting packets. This event reporting packet acts as the RTS packet in MACA, while the beacon packet for next superframe acts as the CTS packet. Like the IFS mechanisms in IEEE 802.11, cluster members compete for different CS time slots according to their event priority. Cluster members with higher event priority compete for the early CS time slots, while those with lower one compete the later CS time slots. Similar to the ATIM window in IEEE 802.11, cluster members turn on the radio for the time slots during CS and turn off the radio for the later time of the superframe.

App-MAC uses the reservation-based protocol to transmit event data. In App-MAC, we design a RS assignment algorithm (e.g., WMPQ) to allocate the time slots according to the application requirements and the event status. WMPQ algorithm provides prioritized event delivery and supports motes and events with fairness, considering the multimodality features of wireless sensor network. WMPQ algorithm considers the combination of performance metrics, e.g., average event delivery latency and inter-event and intra-event fairness and channel utilization efficiency, while Independent Stations Algorithm is mainly to maximize throughput from slot to slot. Unlike Prioritized Regulated Allocation Delay-oriented

Scheduling algorithm in DQRUMA, WMPQ algorithm considers the multimodality feature of event data and try to minimize the delay of variable-length event data which come from several cluster members. WMPQ algorithm leverages the approach to assign weight to event data considering the priority of the event, the waiting time of the event, the sensor node number for this event, and the remaining packets for those event-correlated sensor nodes. WMPQ algorithm uses multiple queues and employs heuristic to assign the RS time slots to the cluster members.

VI. Conclusions

In this paper, we have designed and implemented an application-aware, event-oriented MAC protocol (App-MAC) for multimodality WSNs applications, such as the urban target recognition at intersection to alarm the pedestrian or notify the car drivers. App-MAC leverages the advantages of contention-based and reservation-based MAC protocols to coordinate the channel access, and propose channel contention and reservation algorithms to adaptively allocate the time slots according to application requirements and current events status. Our evaluation results via simulation in TOSSIM and empirical study with Berkeley TelosB motes show that the proposed App-MAC protocol is able to support the prioritized delivery of events, provide inter-event and intra-event fairness, and improve the performance of channel utilization efficiency while reducing energy consumption. App-MAC outperforms three other state-of-the-art MAC protocols, i.e., S-MAC, TDMA, and TRAMA, in term of event delivery latency, event and sensor fairness index, channel utilization efficiency and energy consumption efficiency.

VII. Acknowledgements

We wish to thank the paper reviewers for the helpful comments, which greatly improve the quality of the paper. We also thank Prof. Baochun Li for his valuable comments for the evaluation and presentation of the paper.

References

[1] "Saving lives through advanced vehicle safety technology," USDOT, Intelligent Vehicle Initiative Final Report, Tech. Rep. FHWA-JPO-05-057, Dec. 2005. [Online]. Available: http://www.itsdocs.fhwa.dot.gov/JOPDOCS/REPTS_PR/14153_files/ivi.pdf

[2] "Dedicated short range communications (dsrc) home." [Online]. Available: http://grouper.ieee.org/groups/scc32/dsrc/

[3] K. Arisha, M. Youssef, and M. Younis, "Energy-aware tdma-based mac for sensor networks," in *Proceedings of the IEEE Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT'02)*, May 2002.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*, Nov. 2004.

[5] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*, Nov. 2003.

[6] T. v. Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*, Nov. 2003.

[7] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM'02)*, June 2002.

[8] M. Adamou, I. Lee, and I. Shin, "An energy efficient real-time medium access control protocol for wireless ad-hoc networks," in *Proceedings of the WIP session of IEEE Real-time systems symposium(RTSS'01)*, Dec. 2001.

[9] V. Rajendran, J. Garcia-Luna-Aceves, and K. Obraczka, "An energy-efficient channel access scheduling for sensor networks," in *Proceedings The 5th International Symposium on Wireless Personal Multimedia Communication (WPMC'02)*, Oct. 2002.

[10] J. N. Daigle and M. N. Magalh, "Analysis of packet networks having contention-based reservation with application to gprs," *IEEE/ACM Transactions on Networks*, vol. 11, no. 4, pp. 602–615, 2003.

[11] G. Jolly and M. Younis, "An energy efficient, scalable and collisionless mac layer protocol for wireless sensor networks," *Journal of Wireless Communications and Mobile Computing*, vol. 5, no. 3, pp. 285–304, 2005.

[12] J. Deng, Y. S. Han, and Z. J. Haas, "Analyzing split channel medium access control schemes," *IEEE Transactions on Wireless Communications*, vol. 5, no. 5, pp. 967–971, 2006.

[13] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. A. Brewer, and D. E. Culler, "The emergence of networking abstractions and techniques in tinyos," in *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI'04)*, Mar. 2004.

[14] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05)*, Apr. 2005.

[15] IEEE802.15.4-2006, "Part 15.4: Wireless medium access control(mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans)," Sept. 2006.

[16] IEEE802.11, "Part 11: Wireless lan medium access control (mac) and physical layer (phy) specification," Aug. 1999.

[17] C. Wu and V. Li, "Receiver-initiated busy-tone multiple access in packet radio networks," in *Proceedings of Annual ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'87)*, Oct. 1987.

[18] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "Macaw: A media access protocol for wireless lans," in *Proceedings of Annual ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'94)*, Aug. 1994.

[19] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity*. Dover Publications, 1998.

[20] I. Aad and C. Castelluccia, "Differentiation mechanisms for ieee 802.11," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM'01)*, May 2001.

[21] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in

Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03), Nov. 2003.

[22] A. Chandra, V. GUMMALLA, and J. O. LIMB, "Wireless medium access control protocols," *IEEE Communication Surveys*, vol. 3, no. 2, pp. 2–15, 2000.

[23] S. Kumar, V. S. Raghavan, and J. Deng, "Medium access control protocols for ad hoc wireless networks: a survey," *Elsevier Ad-Hoc Networks Journal*, vol. 4, pp. 326–358, 2006.

[24] I. Demirkol, C. Ersoy, and F. Alagz, "Mac protocols for wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115– 121, 2006.

[25] N. Abramson, "The aloha systemłanother alternative for computer communications," *Proceedings of the Fall Joint Computer Conference*, vol. 37, p. 281C285, 1970.

[26] C. Lau and C. Leung, "A slotted aloha packet radio network with multiple antennas and receivers," *IEEE Transactions on Vehicular Technology*, vol. 39, no. 3, p. 218C226, 1990.

[27] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part iłcarrier sense multiple access modes and their throughput-delay characteristics," *IEEE Transactions on Communication*, vol. 23, no. 3, p. 1400C1416, 1975.

[28] P. Karn, "Maca: a new channel access method for packet radio," in *Proceedings of the 9th ARRL/CRRL Amateur Radio Computer Networking Conference*, Sept. 1990.

[29] C. Fullmer and J. Garcia-Luna-Aceves, "Floor acquisition multiple access (fama) for packet-radio networks," in *Proceedings of Annual ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'95)*, Aug. 1995.

[30] G. Bianchi, F. Borgonovo, L. Fratta, L. Musumeci, and M. ZORZI, "C-prma: the centralized packet reservation multiple access for local wireless communications," *IEEE Transactions on Vehicular Technology*, vol. 46, no. 2, pp. 422–436, 1997.

[31] R. Bolla, F. Davoli, and C. Nobile, "A rra-isa mutliple access protocol with and without simple priority schemes for real-time and data traffic in wireless cellular systems," *Mobile Networks and Applications*, no. 2, pp. 35–53, 1995.

[32] M. J. Karol, Z. Liu, and K. Y. Eng, "An efficient demand-assignment multiple access protocol for wireless packet (atm) networks," *Wireless Networks*, vol. 1, no. 3, pp. 267–79, 1995.

[33] J. Mikkonen, J. Aldis, G. Awater, A. Lunn, and D. Hutchison, "The magic wand - functional overview," *IEEE Selected Areas in Communications*, vol. 16, no. 6, pp. 953–72, 1998.

[34] S. Singh and C. Raghavendra, "Pamas-power aware multi-access protocol with signaling for ad hoc networks," *ACM Computer Communication Review*, vol. 28, no. 3, pp. 5–26, 1998.

[35] E. Jung and N. Vaidya, "A power control mac protocol for ad hoc networks," in *Proceedings of the 8th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'02)*, Sept. 2002.

[36] D. Deng and R. Chang, "A priority scheme for ieee 802.11 dcf access method," *IEICE Transactions on Communications*, vol. 82, no. 1, 1999.

[37] A. Pal, A. Dogan, and F. Ozguner, "Mac layer protocols for real-traffic in ad hoc networks," in *Proceedings of the IEEE International Conference on Parallel Processing (ICPP'02)*, Sept. 2002.

[38] C. Lin and M. Gerla, "Maca/pr: An asynchronous multimedia multihop wireless network," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM'97)*, Mar. 1997.

[39] N. Vaidya, S. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless lan," in *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00)*, Aug. 2000.

[40] L. Bao and J. J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for ad hoc networks," in *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01)*, July 2001.

[41] I. Rhee, A. Warrier, M. Aia, and J. Min, "Zmac: a hybrid mac for wireless sensor networks," in *Proceedings of the 3nd International Conference on Embedded Networked Sensor Systems (SenSys'05)*, Nov. 2005.

[42] I. Rhee, A. Warrier, and L. Xu, "Randomized dining philosophers to tdma scheduling in wireless sensor networks," North Carolina State University, Tech. Rep., Feb. 2004.

[43] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4nd International Conference on Embedded Networked Sensor Systems (SenSys'06)*, Nov. 2006.

[44] G. S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo, "Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks," in *Proceedings of the 4nd International Conference on Embedded Networked Sensor Systems (SenSys'06)*, Nov. 2006.