IEICE TRANS. COMMUN., VOL.E96–B, NO.6 JUNE 2013

PAPER

MAC Protocol for Energy Efficiency and Service Differentiation with High Goodput in Wireless Sensor Networks

SangKwon MOON†a), *Member*, Jong-Woon YOO†, Jaesub KIM†, *Nonmembers*, *and* Kyu-Ho PARK†, *Member*

SUMMARY In the sensor networks for surveillance, the requirements of providing energy efficiency and service differentiation, which is to deliver high-priority packets preferentially, while maintaining high goodput, which is to deliver many packets within their deadline are increasing. However, previous works have difficulties in satisfying the requirements simultaneously. Thus, we propose GES-MAC, which satisfies the requirements simultaneously. GES-MAC reduces idle listening energy consumption by using a duty cycle, periodic *listen* (i.e., turn on radio module) and *sleep* (i.e. turn off radio module) of sensor nodes. *Cluster-based multi-hop scheduling* provides high goodput in a duty-cycled environment by scheduling clusters of nodes in the listen period and opportunistically forwarding data packets in the sleep period. *Priority-aware schedule switching* makes more high-priority packets reach the sink node by letting high-priority packets preempt the schedules of low-priority packets. In experiments with MICA2 based sensor nodes and in simulations, the energy consumption of the radio module is reduced by 70% compared to the approaches without a duty cycle, while providing 80% ∼ 100% goodput of the approaches that provide high goodput. Service differentiation is also supported with little overhead.

key words: *WSN, MAC, goodput, energy efficiency, service differentiation*

1. Introduction

One of the key applications of wireless sensor networks (WSNs) is monitoring a given environment, such as surveillance of enemy invasion in security systems [25]. In WSNs, thousands of battery-powered sensor nodes generate management packets *periodically* to maintain the network or data packets *unexpectedly* when an event is detected. Both types of packets should be delivered to the sink within the given time deadlines, and it is especially critical that the event must not be missed or informed after too long a delay. These WSNs should meet the following three requirements.

- **High goodput support**: In surveillance networks (such as enemy invasion), the sensed data reflect the target status, such as the position, pictures of moving targets or the temperature of a fire. The target status can change easily with time. Thus, the sensed data is only valid for a limited period of time (i.e., *packet deadline* [19], [25]). WSNs should deliver packets to the sink within their deadline (e.g., five seconds) as many as possible. This indicates that providing high *goodput*, which is defined as the number of packets that arrived at the sink node within their deadline during a unit time [14], is an important requirement of WSNs.
- **Service differentiation support**: In such surveillance networks, the importance of a data packet depends on its types [9], [13]. For example, the packets for the detected enemy have a higher priority than routine management packets. WSNs should preferentially deliver high-priority packets.
- **Energy efficiency**: Since sensor nodes are usually deployed in places where people cannot easily access, the nodes should operate energy efficiently to increase the battery lifetime [1], [3].

This paper proposes a new MAC protocol which provides energy efficiency and service differentiation while maintaining high goodput. Before presenting our work, we briefly describe several previous works that are closely related to our work.

SMAC [1] and TMAC [2] first achieve energy efficiency by suggesting the synchronous duty cycle. Duty cycle is the periodic *listen* (i.e. turning on the radio) and *sleep* (turning off the radio) of each node. In the listen period, all sensor nodes activate their radio interfaces and participate in data forwarding. In the sleep period, they turn off the radio interfaces for energy conservation. By SMAC-like duty cycling, each node conserves much energy and prolongs the battery lifetime. However, as data packets can only be forwarded during the listen period, emergent events cannot be notified within a short deadline and the goodput is sacrificed for energy efficiency.

LASMAC [3], [4], RMAC [5], and DW-MAC [12] are energy efficient as well as achieve relatively higher goodput by node *scheduling* in a multi-hop environment. Unlike SMAC, the listen period is used to schedule nodes in a routing path (i.e., deciding when to wake up, *send*, or *receive*) and data packets are forwarded in the sleep period along with the scheduled nodes. Other unscheduled nodes sleep for energy saving. As packets are forwarded in the sleep period, such schedule-based protocols support higher goodput than SMAC. However, because only a single path is scheduled for a data flow, the goodput of the previous schedule-based protocols still suffers from retransmissions occasionally happened on each hop in lossy WSNs [6], [19].

Furthermore, such schedule-based protocols have difficulties in providing service differentiation. In LASMAC and RMAC, once a node is reserved for delivering a low-priority packet, an urgent packet (e.g. a highest-priority packet) that arrives late at the node must wait until the low-

Manuscript received July 12, 2012.
Manuscript revised January 7, 2013.

†The authors are with the Dept. of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Guseong-dong, Yuseong-gu, 305-701 Daejeon, Republic of Korea.
a) E-mail: skmoon@core.kaist.ac.kr

DOI: 10.1587/transcom.E96.B.1444

priority packet goes through according to the schedule. For this problem, previous service differentiation methods [7]–[9] (e.g. varying DIFS and CW size depending on packet priority) can be applied to LASMAC, but increased service differentiation effect is small. This necessitates a totally new design of service differentiation scheme.

To achieve the goals mentioned above, we propose a MAC protocol which supports high Goodput, Energy efficiency, and Service differentiation (GES-MAC). GES-MAC consists of two techniques: *Cluster-based Multi-hop Scheduling* and *Priority-aware Schedule Switching*.

Cluster-based Multi-hop Scheduling (CMS) is designed to maintain energy efficiency by using a duty cycle and to provide high goodput by solving the retransmission problem on each hop. Unlike previous scheduling-based approaches that schedule only nodes in the routing path, our CMS schedules nodes by the cluster, called *scheduling cluster* (SC), which consists of both routing nodes and their one-hop neighbor nodes. CMS enables data packets to be forwarded hop-by-hop in the opportunistic forwarding manner [17], [18] during the sleep period. Then, the number of retransmissions can be dramatically reduced because the probability that at least one node in a cluster will successfully receive the packet is much higher than the probability that only the next routing node successfully will receive the packet, resulting in goodput increase. Our experimental results show that our approach can provide high goodput, although a little more energy is consumed than LASMAC.

Priority-aware Schedule Switching (PaSS) is proposed to support service differentiation in schedule-based WSNs. PaSS allows high-priority packets to preempt the schedules of low-priority packets so that the high-priority packets can be served earlier. Then, even though a node is reserved for a low-priority packet earlier, a high-priority packet is forwarded faster than the low-priority packet, resulting in the higher goodput of high-priority packets. We minimize the overhead in PaSS to avoid goodput degradation. Our experimental results show that PaSS supports service differentiability in schedule-based WSNs with negligible overhead.

Note that each node in GES-MAC manages only its one-hop neighbors' information for scalability. Note that this paper just focuses on the usage of duty cycle to enhance energy efficiency and the balancing of energy utilization among sensor nodes is beyond the scope of this paper. It is because the current version of GES-MAC is implemented on geographic routing [28], but it may be achieved by applying energy-aware routing protocols [26] to GES-MAC.

The contribution of this paper is as follows:

- We analyze the difficulties of providing high goodput in duty cycled environment in WSNs, while providing service differentiation. We propose GES-MAC, a new MAC protocol which satisfies the requirements.
- We validate the protocol operation and evaluate its performance via experiments with MICA2-based Tip30 nodes [30] and simulation.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 presents the GES-MAC. Section 4 discusses the performance evaluation of the proposed protocol. Section 5 concludes the paper.

2. Related Work

Cooperative Opportunistic Routing (COR) [17] provides higher goodput than single path routing (e.g. geographic routing [27]) by using the concept of opportunistic routing [18]. In COR, a data forwarding node broadcasts a data packet, and then, the next node is opportunistically decided among the nodes which have successfully received the DATA packet. In this manner, COR increases goodput compared to single path routing. For example, if five nodes whose link reliability is 50% try to receive a data packet from node *A*, the probability that at least one node will successfully receive the DATA packet is increased to $1 - (1 - 0.5)^5 = 97\%$. This method reduces the number of retransmissions and results in higher goodput. In COR, the radio module of each sensor node is always turned on; thus, much energy is consumed at each sensor node.

To reduce energy consumption, SMAC [1] and TMAC [2] suggest the duty cycle, which means periodic *listen* (i.e., turn on radio module) and *sleep* (i.e., turn off radio module) of sensor nodes. Duty cycle is common feature of energy efficient MAC [3], [5], [6], [10], [15], [16]. As the energy consumption on the radio module is dominant in sensor nodes (more than 60% of total) [1], the energy consumption is reduced. However, in SMAC, data packets are forwarded during only listen period, and data packets can be forwarded just one or two hops in a cycle. Thus, SMAC has goodput degradation problem. For the same reason, A naive combination of SMAC and COR has also the problem.

BMAC [10] and XMAC [11] achieve higher energy efficiency by exploiting shorter listen period (e.g. 1∼5% of a cycle) than SMAC. They also enhance the goodput by sending a data in the sleep period. In BMAC, a node sends a *long preamble packet* for its next node not to sleep in the sleep period, and then, the node transmits the data packet. In this manner, BMAC shows better goodput than SMAC, but the usage of preamble packet induces the delay problem because the data packet goes through the overhead by preamble packets on every hop. Thus, those protocols have difficulties in providing high goodput, especially in a multi-hop environment.

To compensate the goodput degradation caused by the duty cycle, LASMAC [3], [4], RMAC [5], and DW-MAC [12] use the listen period to schedule nodes in a routing path. In the example of Fig. 1, node *A* sends a LAS-RTS packet through routing nodes *B*, *C*, *D*, and *E* in the listen period. Then, the nodes are scheduled to wake up at the specific time of the sleep period with the information of LAS-RTS. In the sleep period, only the scheduled nodes wake up based on their schedule and forward data packet while the other nodes sleep in the sleep period. In this manner, LASMAC and RMAC achieve higher goodput than SMAC while maintaining energy efficiency.

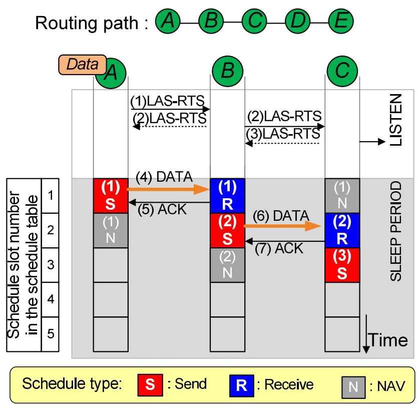


Fig. 1 Operation of LASMAC [3], [4] and RMAC [5].

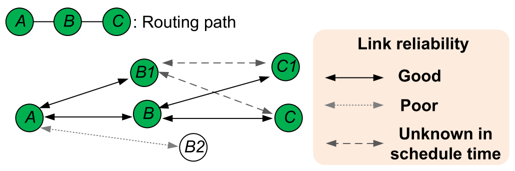


Fig. 2 Problem of naive combination of LASMAC and COR.

However, compared to opportunistic forwarding approaches, such as COR [17], the goodput enhancement is still insufficient. Our experimental results show that the goodput of LASMAC is only 40% that of COR. We analyzed LASMAC and found that the goodput degradation is mainly caused by retransmissions in data forwarding (i.e., in sleep periods). To achieve similar performance to COR without losing the energy efficiency of LASMAC or RMAC, retransmissions should be reduced.

To reduce retransmissions, combining the concept of LASMAC (or RMAC) and COR can be considered. However, combining the concept is not straightforward, because LASMAC needs a routing path, on contrary, COR chooses the best next-hop node on every transmission dynamically. We easily consider a naive combination of LASMAC and COR, which modifying LASMAC to schedule more neighbors (*B1* or *C1*) of each routing node (*B* or *C*) in the listen period and to ask them to be awake, as a solution. However, such a naive combination has a critical problem shown in Fig. 2. During the sleep period, a data packet can be forwarded from node *A* to one of nodes *B* and *B1* opportunistically, (i.e., reliably), but we cannot ensure reliable transmission from node *B1* to the next-hop nodes (*C*, *C1*) because the link qualities among them are not considered in scheduling. In the worst case, nodes *B1* may have no link to the next-hop nodes. Thus, we cannot expect considerable goodput enhancement from the naive combination of LASMAC and COR (just 50% of COR), necessitating the development of a new approach for scheduling and forwarding to create a synergetic effect.

Furthermore, LASMAC and RMAC have difficulties in providing service differentiation. Figure 3 shows the problem of LASMAC. Node *Y* schedules node *D* for Data1 (low-

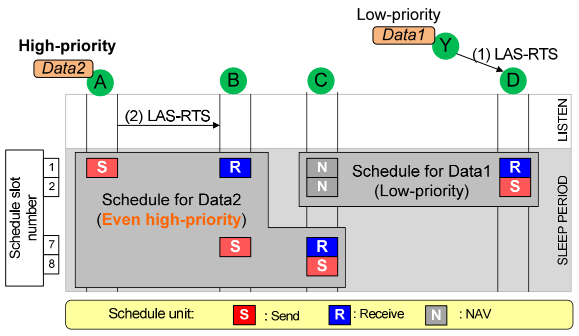

Low-priority
Data1
Y
(1) LAS-RTS
High-priority
Data2
A B C D
(2) LAS-RTS
LISTEN
Schedule slot number
Schedule for Data2 (Even high-priority)
Schedule for Data1 (Low-priority)
SLEEP PERIOD
Schedule unit: S : Send R : Receive N : NAV

Fig. 3 Problem of LASMAC in providing service differentiation.

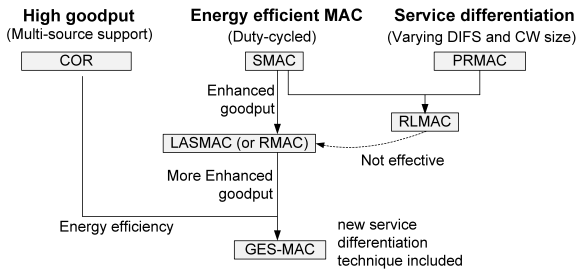


Fig. 4 Evolution of related work.

Table 1 Comparison of related work.

Protocol	Goodput (normalized)	Energy Consumption (normalized)	Service Diff.
COR [17]	**100**	100	NO
RLMAC [9] (SMAC [1] based)	7	**25**	**YES**
RLMAC + COR	13	**25**	**YES**
BMAC [10]	16	**4**	NO
LASMAC [3] RMAC [5]	40	**26**	NO
GES-MAC	**80 ~ 100**	**30**	**YES**

priority) first. After that, if a LAS-RTS packet for *Data2* tries to schedule node *C*, node *C* is already scheduled to NAV. Thus, node *C* is scheduled for *Data2* with time delay, even though the priority of *Data2* is higher than that of *Data1*. In worst cases, even though the same number of high and low-priority packets is generated, the goodput for high-priority packets is lower than that for low-priority packets. Applying previous service differentiation methods [7]–[9] (e.g., by varying DIFS or CW size) is not effective to the problem either. This necessitates the development of a new service differentiation method for our environment.

In summary, the technical advancement flow of the aforementioned works is shown in Fig. 4. Previous works or their simple combinations have difficulties in providing high goodput in duty-cycled environments and in providing service differentiation, as shown in Table 1. To achieve the requirements (i.e., providing energy efficiency and service differentiation while maintaining high goodput), we propose a new MAC protocol, GES-MAC.

3. MAC Protocol for Energy Efficiency and Service Differentiation while Maintaining High Goodput (GES-MAC)

The proposed MAC protocol is designed to meet the following two goals simultaneously:

- Maintaining high goodput in duty-cycled (i.e. energy efficient) environments.
- Supporting service differentiation.

GES-MAC consists of two major techniques. *Cluster-based Multi-hop Scheduling* (CMS) is to maintain high goodput in duty-cycled environments. *Priority-aware Schedule Switching* (PaSS) is to provide service differentiation. Before describing them, we first introduce our assumptions for wireless sensor networks.

3.1 Assumptions about Sensor Networks

Sensor nodes and applications in sensor networks have the following assumptions:

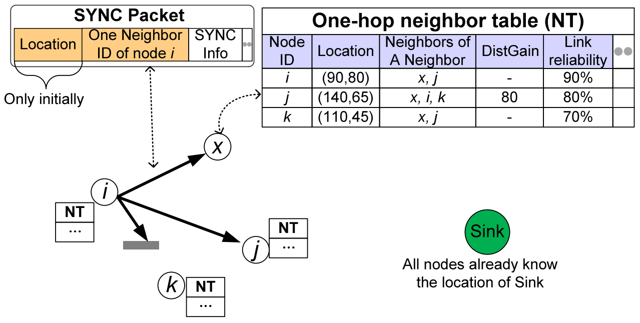
- Each sensor node knows its location. To this end, each sensor node may have GPS modules [27] or may run other location aware algorithms [29]. Each sensor node also knows the location of sink node.
- Large numbers of sensor nodes are densely placed [19] and sensor nodes are almost stationary (i.e., with almost no mobility).
- The size of data packets is fixed.
- The total area is divided into several zones (i.e. group of nodes) and the duty cycle of nodes of each zone is synchronized by using synchronization packets (SYNC packets) [1], [3], [5].

The routing path is set up by a geographic routing [28], which can be better for increasing goodput and works with just neighbor information. Thus, a neighbor node which is closer to the sink and has better link reliability is selected as the next routing node.

3.2 Cluster-Based Multi-Hop Scheduling (CMS)

3.2.1 Overview of CMS

CMS uses a synchronized duty cycle and each cycle is divided into three periods: *Sync* period, *Scheduling* period, and *Forwarding* period. In the *Sync* period, each node wakes up and shares *SYNC* packet in order to synchronize the duty cycle of each node [3], [5], [12] and in order to fill the data structure of each sensor node. In the scheduling period, CMS schedules nodes which are needed for data forwarding by unit of cluster, called Schedule Cluster (SC). In the forwarding period, only scheduled nodes keep awake and join data forwarding, and the other nodes sleep for energy conservation. Unlike previous schedule-based approaches [3], [5], [12], CMS dynamically schedules nodes by unit of SC,



One-hop neighbor table (NT)

Node ID	Location	Neighbors of A Neighbor	DistGain	Link reliability	●●
i	(90,80)	x, j	-	90%	
j	(140,65)	x, i, k	80	80%	
k	(110,45)	x, j	-	70%	

Fig. 5 The SYNC packet and data structure of each sensor node.

which is defined as a routing node and its one-hop neighbor nodes. To this end, CMS manages the one-hop Neighbor Table (NT) and uses a control packet called Cluster-based Multi-hop RTS (CM-RTS) packet. In this section, first, we explain how to manage NT, and then, we briefly explain how to schedule SCs and forward data through SCs.

In CMS, each node manages the one-hop Neighbor Table (NT) which contains the location information, the link reliability, DistGain, and the neighbor nodes of each of its one-hop neighbor nodes, etc. One-hop neighbor nodes are the nodes which communicate directly with one-hop broadcast. The information on NT is easily gathered by using *SYNC* packet. First, the location information of a neighbor node is easily gathered by adding the location information on the *SYNC* packet. In Fig. 5, node i broadcasts its *SYNC* packet with its location information. Then, its neighbors (e.g., node x and node j) easily know the location information of node i. The location information is included in the *SYNC* packet only initially, because nodes have almost no mobility. Second, the link reliability is got by the *SYNC* packet success rate. On the *SYNC* information (SYNC info), *SYNC* sequential number is included. With the *SYNC* sequential number, node x knows the total number of transmitted packets from node i and also keeps counting the number of the successfully received packets. Then, the link reliability of node i is easily measured by the ratio of two values [22], [23]. Third, DistGain is got by calculation. DistGain represents that which neighbor node is closer to the sink node. At node x, the *DistGain* for node j is calculated as

$$DistGain(j) = DistToSink(x) - DistToSink(j), \tag{1}$$

where DistToSink(m) is the distance from node m to the sink node. As the neighbor node whose $DistGain \times Link\ reliability$ is maximum is selected as the next routing node in CMS (i.e. geographic routing [28]), DistGain is managed. Fourth, the neighbor nodes of a neighbor are shared by adding an ID of the broadcaster's neighbor. In Fig. 5, node i adds either the ID of node x or node j at each *SYNC* broadcasting. Then, node x knows node x and node j are all neighbor nodes of node i. In the example of Fig. 5, node k is not a neighbor of node i because of the block. In this manner, each node can manage the NT without big overhead.

In the scheduling period, CMS schedules SCs by for-

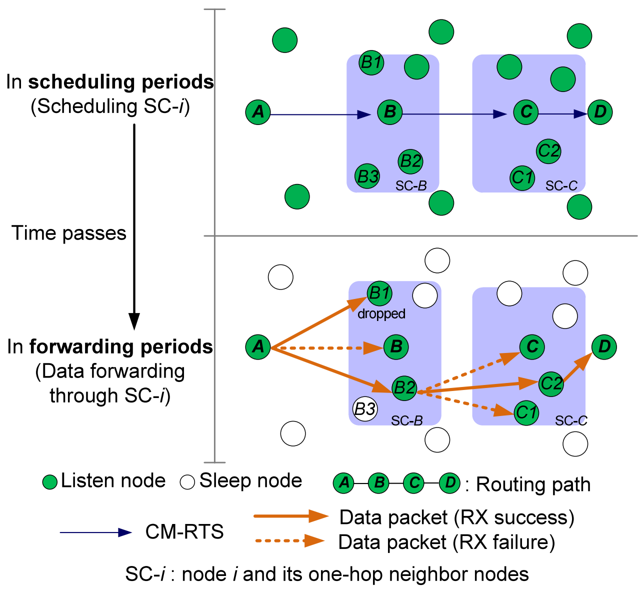


Fig. 6 Overview of CMS.

warding the CM-RTS packet through a routing path. According to the schedule information in the CM-RTS packet, each routing node is scheduled and the neighbor nodes of each routing node are also scheduled by overhearing the CM-RTS packet. In the following forwarding period, the nodes in the same SC will wake up simultaneously according to their schedules. Then, each data forwarding node selects its candidates of next node among SCs dynamically and forwards data packets, allowing data packets to be forwarded opportunistically with a high success rate. For example, in Fig. 6, the routing path is set to *A*-*B*-*C*-*D*, and node *A* has a data packet. Then, in the scheduling period, node *A* sends a CM-RTS packet through node *B*, *C*, and *D*. Based on the scheduling information in the CM-RTS packet, node *B* and node *C* are scheduled. The other nodes in the SCs are scheduled by overhearing the CM-RTS packet. In this example, SC-B (i.e., node *B* and its one-hop neighbors) and SC-C are scheduled. At the beginning of the following forwarding period, node *A* and the nodes in SC-*B* wake up at the same time to send and receive the data packet. Then, node *A* selects some nodes in SC-*B* (e.g., *B*, *B1*, and *B2*) as the forwarder candidates because node *A* knows the neighbors of node *B* (i.e., SC-*B*) with its NT (Fig. 5). Depending on the link condition between the sending and receiving nodes, some nodes (*B1* and *B2* in Fig. 6) may successfully receive the packet and the other (*B* in Fig. 6) may not. Then, CMS chooses one node which is closest to the destination among multiple successful receivers as the next node for data forwarding. In the example shown in Fig. 6, node *B2* is chosen as the new forwarder. Only the chosen node sends the data packet to its candidates of next node, while the others just drop it to avoid contention. The same process is repeated between node *B2* and SC-*C* until the packet reaches the destination. In this manner, CMS forwards data packets through SCs. Just the CM-RTS packet and the information on NT are needed for data forwarding in CMS.

This effectively reduces the number of retransmissions, because of link characteristics. Basically, the transmissions from the sender to multiple receivers are pairwise independent [17], [20]. Thus, in the example of Fig. 6, if the link reliability between node *A* and each node in SC-*B* is 50% each, the probability that at least one node among nodes *B*, *B1*, and *B2* will successfully receive the data packet increases to $1-(1-0.5)^3 = 87.5\%$. In some cases, there can be temporal correlation among some links [24], but the links are independent for most time. Therefore, by arbitrating multiple receivers so that only one of them gets to participate in data forwarding to avoid collision, we can expect high goodput.

As shown in Fig. 6, some nodes in a SC (e.g., node *B3*) do not overhear the CM-RTS packet by temporal fading and shadowing. Then, the nodes do not wake up at the forwarding period. Thus, when a forwarder selects the candidates of new forwarder, the probability about this case should be considered. In Sect. 3.2.4, how to select candidates will be described in detail.

Note that the role of routing nodes in CMS is different from that in previous schedule-based approaches [3]–[5], [12]. In the previous approaches, routing nodes deliver both RTS and data packets through a single routing path. However, in CMS, routing nodes handle only RTS (i.e., CM-RTS) packets to schedule nodes in the scheduling period.

Three major technical issues must be addressed to achieve the high goodput with the CMS approach: (i) *how to schedule SCs efficienty in the scheduling period*? (ii) *how to arbitrate multiple receivers in the forwarding period*? (iii) *how to select the candidates of a new forwarder*? The next subsections discuss these issues with a detailed explanation.

3.2.2 Scheduling SCs in the Scheduling Period

To provide high goodput, CMS needs to schedule both routing nodes and their one-hop neighbors by the unit of scheduling cluster (SC) during short scheduling period effectively. To this end, instead of sending a scheduling packet (i.e., CM-RTS) to every node that will participate in data forwarding, CMS simply forwards the CM-RTS to the destination along the routing path; thus, only the routing nodes are *explicitly* involved in scheduling, as shown in Fig. 6. The neighbors of the routing nodes then overhear the CM-RTS propagated along the routing path and schedule themselves *implicitly* using the information in the overheard CM-RTS.

We will explain the scheduling operation with an example shown in Fig. 7. Each node manages a time table called a *schedule table*, which describes when to wake up, receive, send, and return to sleep for the node. Time is divided by the slot, *schedule slot* and is synchronized for all nodes. Schedule Slot Number (SSN) represents timing during a forwarding period. Figure 8 represents the structure of CM-RTS. For scheduling, it contains two components of scheduling information, *SSN for Send and Receive*, shortly SSN (i, j) in Fig. 7. Then, the scheduling operation is carried out as follows.

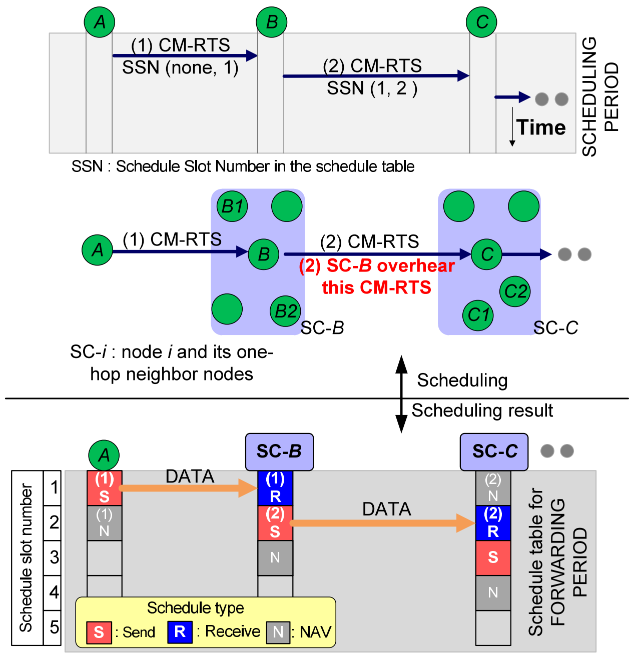


Fig. 7 Scheduling SCs in scheduling period.

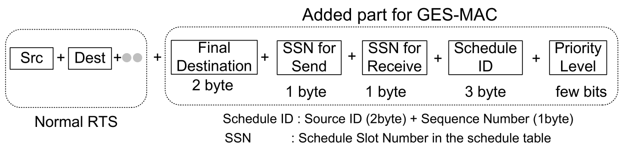


Fig. 8 Cluster-based Multi-hop (CM) RTS packet.

Node *A* sets a *Send* at the first slot of the schedule table, and issues the CM-RTS. After receiving the CM-RTS, node *B* sets a *Receive* at the first schedule slot, in Step (1). Likewise, node *B* sets a *Send* at the second schedule slot and issues the CM-RTS. The *SSN for Send* represents the *Receive* schedule of node *B* and the *SSN for Receive* represents the *Send* schedule of node *B* (i.e., SSN (1, 2)) in Fig. 7. Thus, by overhearing the CM-RTS, the nodes in SC-*B* schedule themselves with the schedule of node *B*. As a result, all nodes in SC-*B* have the same schedule, in Step (2). This scheduling process is repeated until the CM-RTS packet reaches the destination node or the scheduling period expires. In this manner, CMS can schedule nodes by the unit of SC effectively in the scheduling period. This feature is a unique process that distinguishes our CMS from previous approaches.

The schedule of SCs is pipelined by unit of two SCs. For example, in Fig. 7, SC-*B* is scheduled to receive a data packet from node *A*, and SC-*C* is scheduled to receive a data packet from a node in SC-*B*. Nodes in two SCs (e.g., SC-*B* and SC-*C*) choose the earlier schedule as their schedule.

A Schedule ID (SID) is assigned for each schedule. SID consists of the node ID and sequence number. The Priority Level is used for service differentiation in Sect. 3.3.

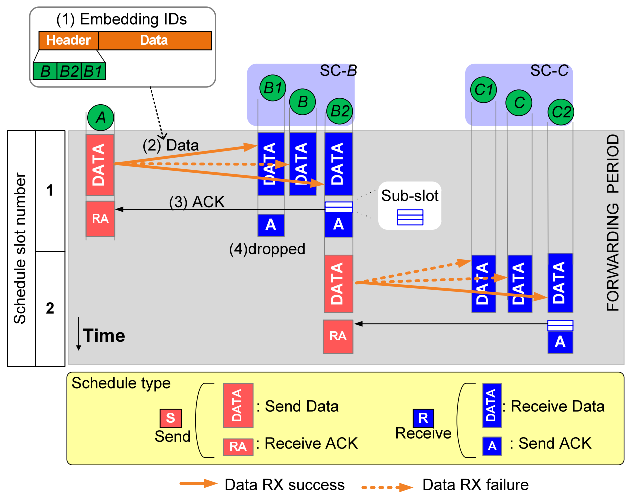


Fig. 9 Data forwarding through SCs in the forwarding period.

3.2.3 Schedule-Based Opportunistic Data Forwarding in the Forwarding Period

As we discussed in Sect. 3.2.1, to provide high goodput by reducing the retransmissions of data packets in lossy environment, CMS allows multiple receivers to simultaneously receive a data packet sent by the previous node. Even though multiple nodes can successfully receive the data packet, only one of them should be chosen as a new forwarder to avoid collision due to simultaneous transmissions. This section describes how to elect new forwarder without collision in schedule-based environment.

CMS prevents collision by embedding a list of new forwarders into the data packet header [18] and dividing the acknowledgment into slots [17]. An example shown in Fig. 9 illustrates the operation of CMS in the forwarding period, according to the schedule of Fig. 7. The schedule types *Send* and *Receive* in Fig. 7 can be divided into *Send Data* and *Receive ACK*, and *Receive Data* and *Send ACK*, respectively, as shown in Fig. 9. CMS forwards data packets in the following sequence:

In Step (1), node *A* selects the candidates of new forwarder among the nodes in SC-*B*, which have high link reliability and are close to the destination. The candidate selection criteria and procedure will be explained later in Sect. 3.2.4. Then, node *A* records the IDs of the candidates in the header of the data packet. The nodes' IDs appear in the order of preference. In CMS, the next routing node (node *B*) is always most preferable. Except routing nodes, a node with a shorter distance to the destination is preferred because such nodes may reduce hops to the destination. As a result, the IDs are embedded to the data packet header in the order of *B*, *B2*, and *B1*, as shown in Fig. 9.

In Step (2), node *A* sends the data packet to SC-*B* at the first schedule. Then, only one new forwarder is decided depending on reception success or reception failure, and depending on the sequence of embedded IDs. The ACK schedule slot is divided into several sub-slots. Then, only the candidates, which have received the data packet successfully, choose the sub-slots by the order of IDs in the CM-RTS. For example, in Fig. 9, *B2* selects the second sub-slot and *B1* selects the third sub-slot. After selecting its sub-slot, *B1* and *B2* wait for the beginning of its sub-slot with overhearing the channel. At the first sub-slot time, there is no carrier because node *B* has failed to receive the data packet. As there was no carrier at the first sub-slot, node *B2* recognizes that node *B2* is the new forwarder. Thus, node *B2* transmits ACK at its sub-slot, in Step (3). By sensing the carrier of the ACK, node *B1* recognizes that there is more preferred node (node *B2*) which has successfully received the data packet. Thus, node *B1* just drops the data packet, in Step (4). As a result, only node *B2* sends the data packet to the next SC, SC-*C*. The above process is repeated until the packet reaches the destination. In this manner, the data packet is forwarded from an SC to another SC without collision and packet duplication.

If a node failed to receive data at its *Receive* schedule, like node *B* in Step (2), it keeps listening to the channel to identify whether other scheduled nodes successfully received the packet or not. If it overhears an ACK sent by another node (node *B2* in Step (3)), it can ensure that the packet is successfully delivered to the next node according to the schedule, and thus, it can go to sleep. However, if the overheard ACK is sent by a node (node *B1*) whose distance to the sink is longer than that of this receive-failed node, it backs off its *Receive* schedule by one schedule slot (i.e., it continues to listen at the next slot) because it can be selected as a next candidate of the node that has sent the ACK. On the other hand, if the ACK is sent by a node closer to the sink (node *B2*), the receive-failed node immediately goes to sleep as it cannot contribute to the delivery of the packet anymore.

In some rare cases, all candidates in a SC fail to receive the data packet. In that case, each SC is rescheduled without additional packet transmission and the rescheduling is done by checking if a node is selected as the forwarder candidate or if the node receives the retransmitted data packet. Figure 10 shows an example of rescheduling. When node *A* has transmitted a data packet, all forwarder candidates (i.e., node *B*, *B1*, and *B2*) fail to receive the data packet. Then, node *A* backs off its schedule to retransmit the data packet because node *A* has not received the ACK packet from any of the candidates. Simultaneously, SC-*B* including forwarder candidates back off their schedule by one schedule slot to receive the retransmitted packet. After this rescheduling, node *A* retransmits the data packet to SC-*B*. In this case, as shown in Fig. 10(b), SC-*B* and SC-*C* wake up simultaneously. However, it does not become a problem because SC-*C* will reschedule themselves. First, because most nodes in SC-*C* will fail to receive the data packet, the nodes will be rescheduled. Second, even though some nodes in SC-*C* luckily have received the data packet, the nodes have not been selected as the forwarder candidates. Thus, the nodes also reschedule themselves, as shown in Fig. 10(c).

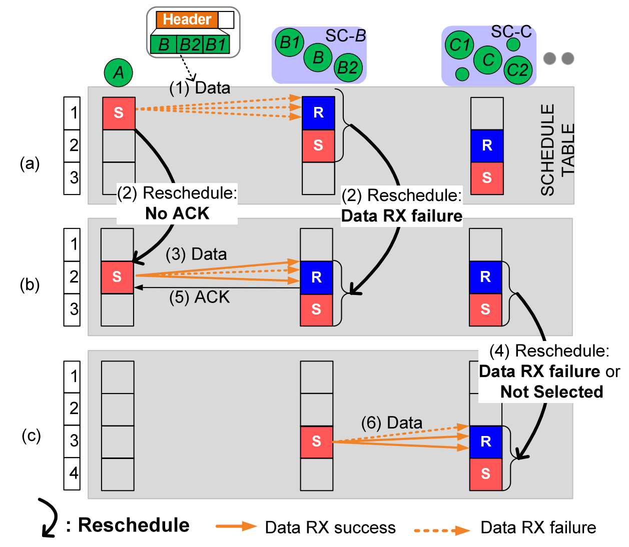


Fig. 10 The rescheduling of SCs for retransmission (schedule back off).

In the same manner, the SCs behind SC-*C* (e.g. SC-*D*) are rescheduled and the data packet is forwarded with just one schedule slot backing off.

Likewise, the ACK packet from node *B2* can be lost, in Step (3). In this case, if node *A* retransmits the data packet, that will be a cause of duplicated transmission. To prevent such cases, in CMS, first, node *A* tries to receive the header of the data packet from node *B2* and uses it as the ACK from node *B2*. Then, the ACK loss probability is decreased much. Rarely, if the header is also lost, node *A* waits until the next listen period and sends a CM-RTS to node *B* to set a schedule for the data packet. However, node *B* knows that the data packet is already forwarded because it has overheard the ACK from node *B2*. Thus, node *B* sends a SCHEDULE-PROCESSED packet to node *A*, and then, node *A* aborts the retransmission and just drops the data packet. In this manner, CMS can prevent the duplication by an ACK loss. Additionally, for the CM-RTS loss, it is just retransmitted like previous works.

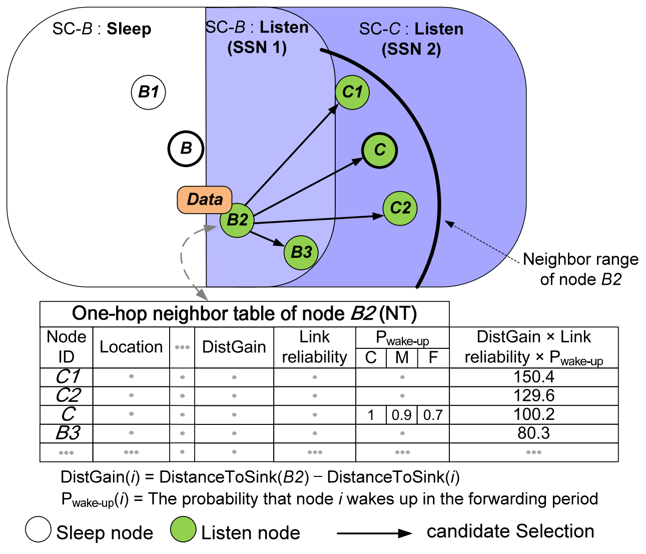
For high goodput, limitation of the number of candidates is necessary because the overhead by selecting a candidate is increased linearly for a selected candidate, while the probability that at least one candidate will successfully receive the data packet is saturated. For example, when the data size is 45 bytes, until six candidates, the goodput is increased as the number of selected candidates is increased. However, if more candidates are selected, the goodput is slightly decreased. If data frame size is changed, the proper number of candidates is also changed.

3.2.4 How to Select the Candidates of a New Data Forwarder

As mentioned in Sect. 3.2.3, in forwarding periods, a node that has data packets to forward, i.e., data forwarder, dynamically selects a fixed number of nodes, e.g. six nodes, among its neighbors as the candidates of new data forwarder before it transmits the data packet. Similar to [19], [28], nodes which are closer to the sink and have better link reliability are preferred for the candidates as these nodes will possibly reduce the hops to the sink and the number of retransmissions on each hop. In other words, nodes with higher *DistGain* (Eq. 1) × *link reliability* are likely to be chosen as the new data forwarder. As each node manages proper information about its neighbor nodes (i.e., the location information, link reliability, DistGain, as mentioned in Sect. 3.2.1), the forwarder candidate selection is easily done. However, unlike such previous approaches in where all neighbors stay awake when a node transmits data, in our CMS, only the nodes scheduled in the scheduling period will wake up in the following forwarding period. Therefore, the data forwarder should know which of its neighbors are scheduled to receive the packet when it transmits to select advantageous new forwarder candidates properly. For this end, we devise a low-overhead scheduled-node estimation mechanism described in this section.

As described in Sect. 3.2.2, because nodes are scheduled by overhearing the CM-RTS packets forwarded along with the routing nodes, only the nodes that successfully overheard the CM-RTS packets will wake up at corresponding schedules. Even though the probability that a neighbor of a routing node successfully overhears the CM-RTS sent by the routing node would be high because of the small size of the CM-RTS packet [20], but the probability would not be 100% for all neighbors in real wireless environment. The probability would vary depending on several conditions, such as the distance between the sender and receiver, the presence of obstacles between them, or unpredictable interference. Thus, it is important for the current data forwarder to know the link reliability between the corresponding routing node and the routing node's neighbors to expect the success probability of overhearing CM-RTS packets. For this end, one can simply make each neighbor of a routing node share the link reliability by sending (possibly broadcasting) control packets containing all link information, but this would increase control overhead significantly, destroying the system scalability.

The CMS addresses this issue by estimating the probability that a node successfully overhears the CM-RTS packet sent by its neighboring routing node, say $P_{wake-up}$, with the distance between the node and the routing node, instead of exchanging all link information. The each node classifies its neighbors into three groups, close, mid-range, and far, by distance (D). For example, a neighbor with D less than 8 meters goes to the close group. In case of $8 \leq D < 16$ or $16 \leq D < 32$, the node goes to the mid-range or far groups, respectively. The link reliability of each neighbor group is averaged. Each node embeds these three averaged link reliability values into the SYNC packet. The values are continuously saved as Close(C), Middle(M), and Far(F), of $P_{wake-up}$ on the NT, as shown in Fig. 11. Then, when a routing node is decided, the current data forwarder easily finds the $P_{wake-up}$ of a neighbor node with the distance of the neighbor node to the routing node.



One-hop neighbor table of node *B2* (NT)

Node ID	Location	•••	DistGain	Link reliability	$P_{wake-up}$			DistGain × Link reliability × $P_{wake-up}$
					C	M	F	
C1	•	•	•	•	•			150.4
C2	•	•	•	•	•			129.6
C	•	•	•	•	1	0.9	0.7	100.2
B3	•	•	•	•	•			80.3
•••	•••	•	•	•••	•••			•••

DistGain(i) = DistanceToSink($B2$) − DistanceToSink(i)

$P_{wake-up}(i)$ = The probability that node i wakes up in the forwarding period

Fig. 11 An example of selecting the candidates of a new data forwarder.

Then, in CMS, the current data forwarder selects new forwarder candidates among its neighbors using DistGain, link reliability, and $P_{wake-up}$. In the example of Fig. 11, the current data forwarder, node *B2*, selects top six nodes whose DistGain × link reliability × $P_{wake-up}$ values are high. Note that node *B3* which is involved in SC-*B* but failed to receive data in the previous Receive schedule also can be chosen because it is closer to the sink than the current data forwarder (*B2*), as already described in Sect. 3.2.3.

3.2.5 Prescheduling for Periodic Packets

As CMS uses a synchronized duty cycle, a data packet which is generated during a forwarding period should wait until the next scheduling period. The delay, called initial delay, is also the cause of goodput degradation.

However, *prescheduling* removes the initial delay and increases goodput for packets for which a source node can estimate the timing of the next packet generation (e.g., periodic packets). For example, node *A*, a source node, knows the estimated timing of next packet generation in the next forwarding period. Then, node *A* preschedules SCs for the expected packet in this scheduling period, even though node *A* has no packet yet. Then, the data packet can be forwarded without initial delay because the schedule of the data packet has been already set up by prescheduling. Then, initial delay can be removed except the very first packet of periodic packets.

SMAC or BMAC may remove the initial delay by prescheduling the entire routing path, but it is not easy for them to adapt routing path change. For example, when the last data packet has prescheduled the entire routing nodes for a data packet which will be generated ten minutes later, even though the routing node is changed because of poor link or link broken, newly generated periodic packet should travel the old routing path. However, in CMS, just the source node removes the initial delay irrespective of routing path change.

Thus, the *prescheduling* of CMS would be a little more useful for the applications can notify the packet generation time [25].

For randomly populated packets, initial delay is inevitable. Thus, the goodput result for periodic packets is better than that for randomly populated packets.

3.3 Priority-aware Schedule Switching (PaSS)

The primary goal of PaSS is to provide service differentiation in schedule-based wireless sensor networks (WSNs). Here, service differentiation support means that increasing the goodput of high-priority packets as many as possible by forwarding high-priority packets preferentially. In achieving the purpose, the major obstacle is the cases that some SCs are already scheduled for a low-priority packet. Previous approaches are not effective in solving the problem cases or result in total goodput degradation, as mentioned in Sect. 2. In this section, we propose a new method that solves the obstacle without much total goodput degradation.

To achieve service differentiation, PaSS allows high-priority packets to preempt the schedules of low-priority packets so that the high-priority packets can be served earlier. Then, the goodput of high-priority can be increased. For easy understanding, we describe the PaSS protocol using the example shown in Fig. 12. For the sake of simplicity, let us assume that only two levels of priority, *high* and *low*, are supported by PaSS. Therefore, one bit extension in the packet header (Fig. 8) is enough to represent the priority of the packet. The number of priority levels can be easily adjusted by varying the number of priority bits. The operation of PaSS is as follows:

In the scheduling period: The obstacle case is shown in Fig. 12(a). The SC-*D* is already scheduled by *Data1* (low-priority), and *Data2* (high-priority) wants to schedule SC-*D* at a similar time line. In this example, the CM-RTS packet for *Data1* has already passed through nodes of (*A*, *B*, *C*, and *D*). The CM-RTS packet contains the priority of *Data1* (i.e., low-priority) because the priority of *Data1* is embedded to the CM-RTS packet, as shown in Fig. 8. As node *Y* is a neighbor node of node *D*, node *Y* has overheard the CM-RTS packet. Thus, node *Y* knows that node *D* is already scheduled by *Data1* whose priority is low. Node *Y* also knows that the priority of *Data2* is higher than that of *Data1* because it is the sender. Then, node *Y* sends a Schedule Switch Request (SSR) to node *D* and preempts the schedule of *Data1*, in Step (1). As *Data2* preempts the schedule for *Data1*, there is no schedule for *Data1*. To set new schedule for *Data1*, node *D* sends a *RESchedule* (RES) packet to node *C*, in Step (2). By overhearing this RES packet, node *Y* is acknowledged that the schedule preemption is confirmed. Then, node *C* sends a CM-RTS to set another schedule for *Data1* and the new schedule is made with time gap to avoid collision with the schedule of *Data1*, in Step (3). Parts of SC-*B* nodes, which have overheard the CM-RTS, also reschedule the schedule of *Data1* to avoid collision with that of *Data2*. By the new CM-RTS packet, the priority level of the schedule for *Data2* is also updated to high-priority, for the preempted schedule by *Data2* not to be preempted again by another data packet. In this manner, even though there is a schedule for low-priority packet, the high-priority packet is scheduled ahead. For simple explanation, we set node *Y* to a data packet sender, but even though a CM-RTS packet is forwarded from another node to node *Y*, the schedule is preempted in the same manner.

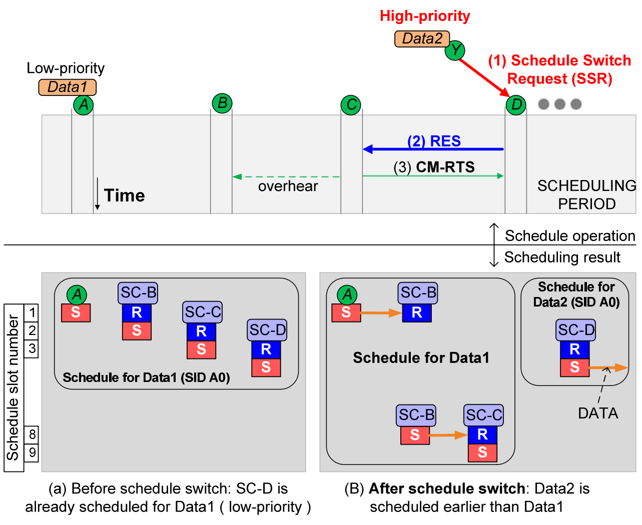

High-priority
Data2
Y
(1) Schedule Switch Request (SSR)
Low-priority
Data1
A
B
C
D
(2) RES
(3) CM-RTS
overhear
Time
SCHEDULING PERIOD
Schedule operation
Scheduling result
Schedule slot number
1
2
3
8
9
SC-B
SC-C
SC-D
R
S
Schedule for Data1 (SID A0)
Schedule for Data1
Schedule for Data2 (SID A0)
DATA
(a) Before schedule switch: SC-D is already scheduled for Data1 (low-priority)
(B) After schedule switch: Data2 is scheduled earlier than Data1

Fig. 12 Example of Priority-aware Schedule Switching (PaSS).

In the forwarding period: Data packets are forwarded based on the schedule in Fig. 12(b). *Data1* from node *A* stops at the forwarder in SC-*B*, and waits until *Data2* is forwarded through SC-*D*. After that, the forwarder in SC-*B* transmits *Data1* to SC-*C*. Each data packet contains the SID which it has to use. Thus, node *A* transmits *Data1* with SID *A0*, but the forwarder in SC-*B* changes the SID to *A50* and transmits *Data1*. Then, the high-priority packet is forwarded earlier based on the schedule. Additionally, if *Data1* and *Data2* are of equal priority, *Data2* is forwarded after *Data1* is forwarded as in other approaches [3], [4], [12].

In this manner, PaSS solves the obstacle in Fig. 12(a) and the goodput of high-priority packets are increased. Moreover, the overhead of PaSS is small.

The overhead for a schedule switch, just two additional packets, a Schedule Switch Request (SSR) and a RESchedule packet (RES), are additionally needed. As CM-RTS packets pass scores of hops during a scheduling period, the overhead for schedule switching (i.e., 2 hops for SSR and RES) is relatively small. In the simulation, the goodput of high-priority packets are clearly increased by PaSS, but the amount of total goodput degradation by PaSS is just 4%.

Additionally, for the loss of new control packets (e.g., SSR and RES), the retransmission is also used like CM-RTS loss. For example, if node *C* fails to receive the RES packet from node *D*, there would be no new CM-RTS transmission from node *C*. In that case, node *D* retransmits the RES packet after time delay.

4. Experimental Results

4.1 Simulation and Experiment Setup

We implemented our GES-MAC as well as previous protocols, namely, LASMAC [3], [4], SMAC [1], BMAC [10], and COR [17] using Tip30 nodes and using NS-2 version 2.33 for performance comparison. Except COR which does not use duty cycling, all other implemented protocols have duty cycle, but the ratio is different. GES-MAC, LASMAC, and SMAC have the same 1:4 ratio of a listen period (i.e., scheduling period in GES-MAC) and a cycle, which means the duty cycle is 25%. The lengths of the listen period and cycle were set to 0.5 and 2.0 seconds, respectively. In BMAC, the lengths of the listen period and cycle were set to 2 ms and 200 ms, which means that the duty cycle is 1%. We also implemented a naive combination of COR and SMAC (or RLMAC [9]), so-called SMAC+COR, to examine whether SMAC+COR can achieve our goals (described in Sect. 2) simultaneously. In SMAC+COR, during the listen period, data packets are opportunistically forwarded as in COR, and all nodes sleep during the sleep period.

In order to test the protocols in a multi-hop environment of multiple sources, the experiment should be done in a wide area with many sensor nodes (e.g. one thousand sensor nodes). However, it is very difficult to deploy one thousand sensor nodes. Thus, we simulated the protocols on NS-2 for the environment with one thousand sensor nodes of multiple sources. We implemented the protocols on Tip30 based on MICA2 for the environment of one or two sources.

In simulations, we use the log-distance path loss model [20], [21] as a channel model. Transmitting power (P_t) is 0 dBm, reference distance (d_0) is 11 m, Tx range is 33 m, path loss at d_0 ($PL(d_0)$) is 55 dBm, noise floor (P_n) is -115 dBm, path loss exponent (n) is 4, and variance(σ) is 4. The randomness of the packet reception rate depending on the distance between sender and receiver was also considered. Channel errors and packet collisions are modeled based on ns-2. To test a multi-source environment, 1000 nodes are regularly deployed in 250 m $\times$ 250 m and the sink is located at (230,230). The row gap and column gap are 8 m each. The distance from each source node to the sink node is larger than 150 m. The *packet deadline* is five seconds.

In tests with Tip30, thirty Tip30 nodes were arranged in 2 m $\times$ 40 m as shown in Fig. 13(b). Tip30 nodes are arranged in three rows and ten columns. The row gap is 1 m and the column gap is 4 m. The source is located at the first column and the destination is placed at the tenth column. The number of sources is one or two. The *packet deadline* is from one second to five seconds. Tip30 uses CC1000 transceiver and uses 915 MHz, in Fig. 13(a). The other simulation and test parameters are summarized in Table 2.

For a routing protocol, we use a geographic routing [28]. The neighbor node whose *DistGain* (mentioned in Sect. 3.2.1) $\times$ *Reliability* is maximum is selected as the next routing node.

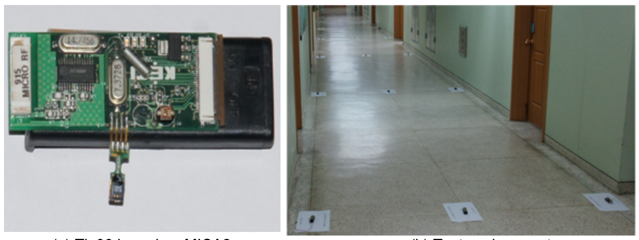

(a) Tip30 based on MICA2 (b) Test environment

Fig. 13 Tip30 [30] (MICA2 platform) and test environment.

Table 2 Simulation and Tip30 test [30] environment settings [20].

Parameter	Values	Parameter	Value
Terrain	Sim. 250 m $\times$ 250 m Expt. 2 m $\times$ 40 m	Node number	Sim. 1000 Expt. 30
Bandwidth	20 kbps	Retry limit	5
Time slot	1 ms	CW	32 slots
SIFS	5 ms	DIFS	10 ms
sub-slot	2 ms	*TXPower*	17 mW
RXPower	15 mW	*IdlePower*	14 mW

The size of the RTS/CTS/DATA/ACK of each protocol is a little different. In SMAC, RTS/CTS/DATA/ACK/SYNC is 10 bytes/10 bytes/45 bytes/10 bytes/9 bytes each. In COR, the RTS packet piggybacks the IDs of selected candidates. Thus, the size of an RTS packet is 20 bytes. In GES-MAC, data packets piggyback the information about selected next nodes. Data packets also carry a schedule ID. Thus, the size of the data packet is 13 bytes larger than that of other approaches. In BMAC, the preamble packet whose size is 271 bytes is used.

We explain the simulation results for multiple sources. After that, we explain the test results with TIP30 for one or two sources.

4.2 Simulation Results with NS-2 Simulator

4.2.1 Maximum Goodput and Energy Consumption

We evaluated the goodput performance and energy consumption of GES-MAC and compared it with COR and previous energy efficient approaches. We also examine whether SMAC+COR can satisfy the goals. Each source generates packets every ten seconds, whose time deadline is set to five seconds. Figure 14 compares the maximum goodput and energy consumption of various protocols.

The goodput of COR is high and that can be regarded as the upper bound, but COR consumes much energy because all nodes keep turning on their radio module, as shown in Fig. 14. On contrary, previous energy efficient protocols consumes less energy, but their goodput are small. SMAC consumes just 25% energy of COR, but the goodput of SMAC is only 7% that of COR because data forwarding only takes place during the listen period, which is under 25% of the total time in our tests. For the same reason, the goodput of SMAC+COR is just 13% that of COR even though SMAC+COR reduces the retransmissions. LAS-

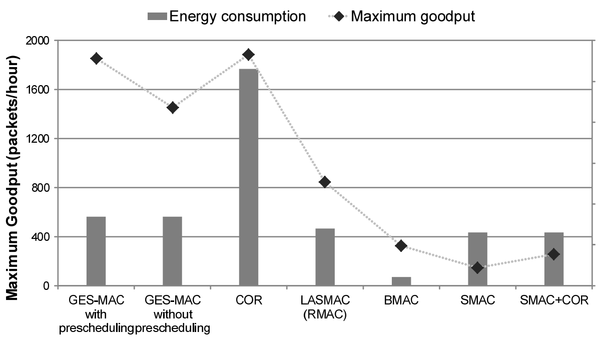


Fig. 14 Maximum goodput and energy consumption.

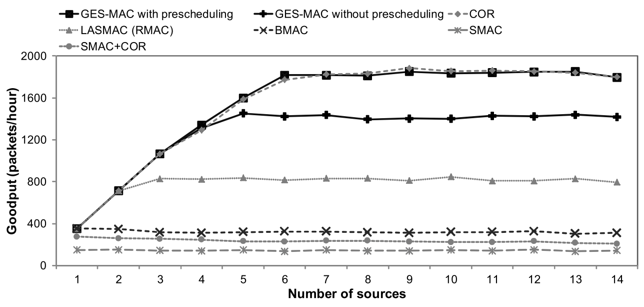


Fig. 15 Goodput in the multi-source environment.

MAC overcomes the goodput degradation of SMAC by node scheduling. However, the goodput of LASMAC is just 40% that of COR because of retransmissions on each hop. BMAC only consumes 4% energy of COR because it has very short listen period, but the goodput of BMAC is also only 16% that of COR because a long preamble packet should be transmitted before sending a data packet on every hop. In the surveillance networks, the goodput is the most important factor. Thus, even though they are energy efficient, previous energy efficient protocols are not proper for such networks.

Although GES-MAC consumes more energy than other energy efficient protocols, GES-MAC reduces much energy than COR, while maintaining high goodput. By Cluster-based Multi-hop Scheduling (CMS) in GES-MAC, more nodes are awake in the sleep time (i.e., forwarding time in GES-MAC) resulting in more energy consumption than LASMAC. The energy consumption of GES-MAC is still just 30% that of COR. However, CMS effectively reduces the number of retransmissions and GES-MAC achieves about 80% of COR's goodput, even in the case that the inter-packet interval is unknown (i.e., prescheduling is not applicable). By analysis, we found that this 20% performance gap is mainly due to the initial delay of the packets which are generated in the middle of the sleep period, as we discussed in Sect. 3.2.5; the other overhead of GES-MAC by larger data packet size and SYNC packet sharing is compensated because GES-MAC has no CTS overhead and forwards a CM-RTS packet with just SIFS (5 ms) delay (in COR, DIFS (10 ms) + random delay (max:32 ms) is needed at each hop). COR does not suffer from initial delays because it immediately begins transmission as soon as a packet is generated. If GES-MAC knows the inter-packet interval, it can enable its prescheduling function. As shown in Fig. 14, when the overhead due to the initial delays is alleviated by prescheduling, the goodput of GES-MAC almost reaches the upper bound. Therefore, GES-MAC is most suitable for WSNs featuring periodic traffic patterns [13], [25]. GES-MAC is also applicable to general WSNs that require a balance of high goodput, energy efficiency.

The result in Fig. 14 shows that GES-MAC consumes more energy than LASMAC and SMAC because more nodes are awakened, but the increased energy consumption is not large. Among 1000 nodes, in SMAC, no nodes are awake during a sleep period. In LASMAC, 14 nodes through routing path are awake during the scheduled time of the sleep period. In GES-MAC, 195 nodes in SCs are awake during the scheduled time of the sleep period (i.e. forwarding period). 805 nodes always sleep in the sleep period. 195 nodes would be awake only during the scheduled time when there are packets to forward. Thus, the amount of increased energy of GES-MAC is not large. As a result, in our simulation, GES-MAC consumes 15% more energy than LASMAC. In the test with Tip30, we have similar results. We will discuss this in Sect. 4.3.2.

We also measured the goodput of each protocol by increasing the number of sources and the result is shown in Fig. 15. As mentioned before, the maximum goodput of GES-MAC and COR is higher than that of BMAC, SMAC, and LASMAC. Like COR, the goodput of GES-MAC does not decrease as the number of sources increases. Therefore, GES-MAC can be used in a multi-source environment.

The simulation results demonstrate that GES-MAC can achieve the goal of providing energy efficiency while maintaining high goodput. In the following section, we will discuss the GES-MAC optimization depending on the target environment.

4.2.2 Discussion about the Optimization of GES-MAC

As mentioned in Sect. 3.2.3, GES-MAC fixes the number of new forwarder candidates which can maximize the goodput, but the number needs to be changed by the size of target frame. In Fig. 16, when data frame is 45 bytes, the goodput is increased until six forwarder candidates are selected because the benefit by reducing the retransmissions and the number of hops which a data packet goes through is higher than the overhead of adding one ID to data packet header. However, after seven candidates are selected, the overhead is a little higher. Thus, the goodput is slightly decreased. If the data frame size is changed, the number of candidates for peak goodput is a little changed. In case of 90 bytes, seven is best. In case of 135 bytes, nine is best.

The duty cycle of GES-MAC must be carefully chosen by the data frame size since it seriously affects both goodput and energy consumption. Figure 17 shows the peak goodput performance of GES-MAC with various duty cy-

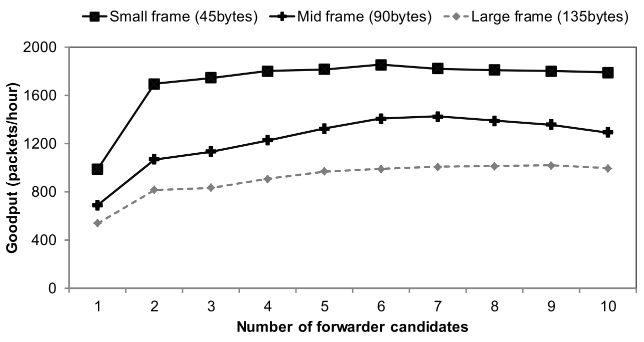


Fig. 16 Number of forwarder candidates for maximizing goodput on the different frame.

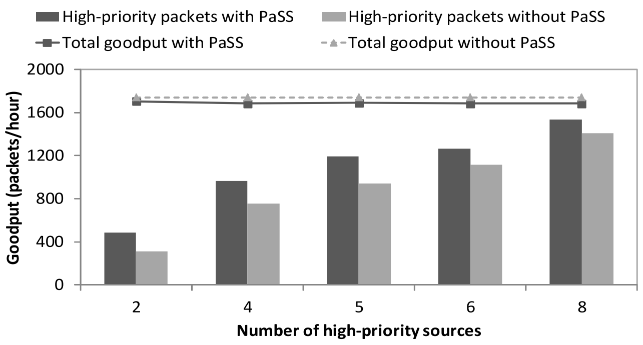


Fig. 18 Service differentiation effect on the different number of high-priority sources.

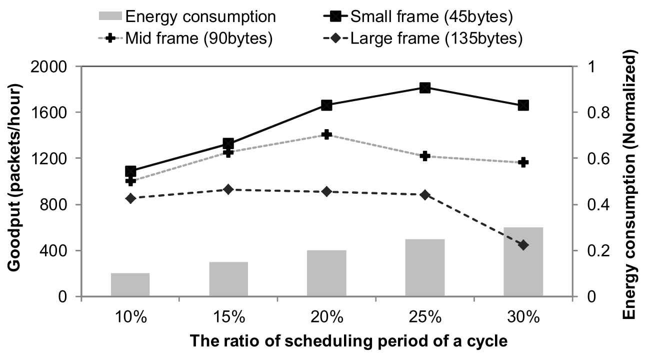


Fig. 17 Duty cycle for maximizing goodput on the different frame size.

cles and data frame size. In Fig. 17, the scheduling period is fixed to 500 ms and the forwarding period is changed by the duty cycle. For example, if the ratio of scheduling period is 25%, the forwarding period is 2 seconds, and if the ratio of scheduling period is 20%, the forwarding period is 2.5 seconds. The energy consumption is normalized to the energy consumed by an always-on node. As the ratio of the scheduling period increases, of course, nodes drain more energy. However, interestingly, the goodput does not increase proportionally to energy consumption. In case of 45 bytes data frame, the goodput of GES-MAC reaches its peak when the ratio is 25% and begins to decline as the scheduling period is lengthened or shortened. This is because of the schedule-based forwarding nature of GES-MAC. In 25% ratio, because the hops which can be scheduled in a scheduling period (e.g., 24 hops) are similar to the hops which can be forwarded in a forwarding period (e.g. 25 hops), the goodput is maximized. If the ratio is smaller (10%), the hops which can be forwarded in a forwarding period (e.g. 100 hops) is larger than the scheduled hops, will cause a large transmission delay and result in goodput degradation. If the ratio is larger (30%), a data packet can travel only 20 hops in a forwarding period, also resulting in goodput degradation. The ratio of scheduling period for peak performance is changed by data frame size because the hops can be forwarded during the forwarding period is changed by data frame size. Thus, in case of 90 bytes frame, 20% shows best goodput. In case of 135 bytes, 15% shows best goodput.

4.2.3 Simulation Results for Service Differentiation

Until now, we have evaluated providing energy efficiency while maintaining high goodput. In this section, we evaluate the service differentiability of GES-MAC and its overhead. Increased goodput of high-priority packets represents better service differentiability. To check this, we divide the given set of sources into two categories, *high-priority* and *low-priority* sources. High-priority sources generate high-priority packets every ten seconds and low-priority sources generate low-priority packets. There are total ten sources. Therefore, the numbers of the high-priority sources reflects the numbers of the high-priority packets in the network. We vary the number of high-priority packets by changing the number of high-priority sources and measured the effects of the Priority-aware Schedule Switching (PaSS) on the goodput of high-priority packets.

We measured the service differentiability of GES-MAC by changing the ratio of high-priority sources. The number of high-priority sources is changed from two to eight among ten sources. Sources are not close one another. The result is shown in Fig. 18. When PaSS is applied, even though the total achieved goodput is slightly decreased, the goodput of high-priority packets is increased (i.e. service differentiation is supported). This is because PaSS allows high-priority packets to preempt the schedules of low-priority packets when a high-priority packet should pass a path reserved by other low-priority packets. The slight degradation (4% on average) of goodput is due to additional packet (e.g., SSR or RES in Sect. 3.3) exchanges for schedule switches. We believe that this overhead is reasonable for the service differentiation support in schedule-based WSNs.

Figure 19 presents a closer look at the effect of the ratio of high-priority packets on the schedule switch count. It shows that the schedule switch count increases if there are similar numbers of high and low priority packets (i.e., five high-priority sources among ten sources). Thus, the increased goodput of high-priority packets is maximized at that point. However, if either of the number of high-priority or low-priority packets is larger, the schedule switch count decreases because schedule switching does not occur be-

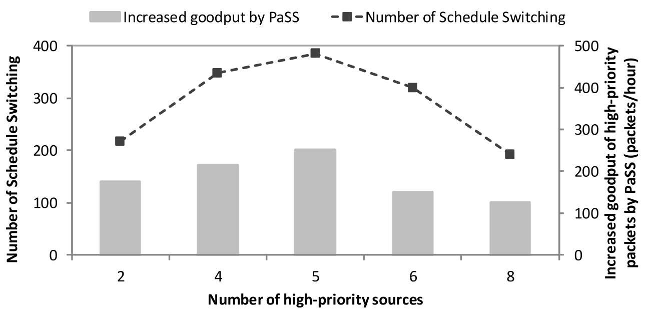


Fig. 19 Number of schedule switch and the high-priority goodput increasing.

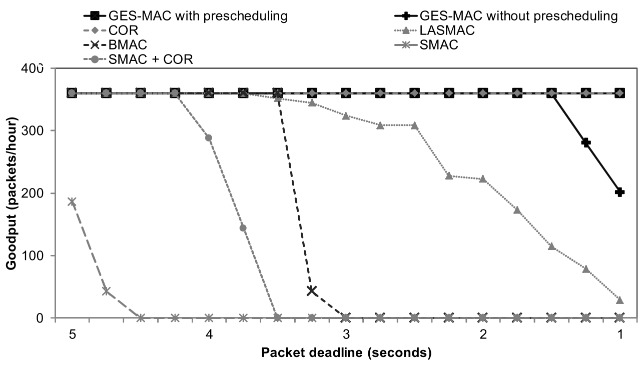


Fig. 20 Goodput with different packet deadline (one source).

tween the packets of the same priority.

4.3 Experimental Results with Tip30

4.3.1 Goodput Performance

The simulation result in Fig. 15 shows that GES-MAC provides high goodput. In the test with Tip30, we also had checked this because the feature also influences supporting short packet deadline. We measured the goodput of each protocol by changing the deadline of the packets. Figure 20 shows the result when there is only one source. If packets whose deadline are 5 seconds are generated, all protocols except SMAC provide high goodput. However, the number of protocols which provides high goodput are reduced as the packet deadline is shortened. If packets whose deadline is 3 seconds are generated, the goodput of BMAC and SMAC+COR become zero. If packets whose deadline is 1 second are generated, the goodput of LASMAC becomes low. Not like LASMAC, even in the case of 1 second deadline, if the packet is generated periodically, GES-MAC can provide high goodput like COR. If the packet is generated randomly, the influence by the initial delay overhead, discussed in Sect. 3.2.5, is big because of short packet deadline. Thus, the goodput of GES-MAC is less than that of COR, but GES-MAC still provides higher goodput than other energy efficient protocols.

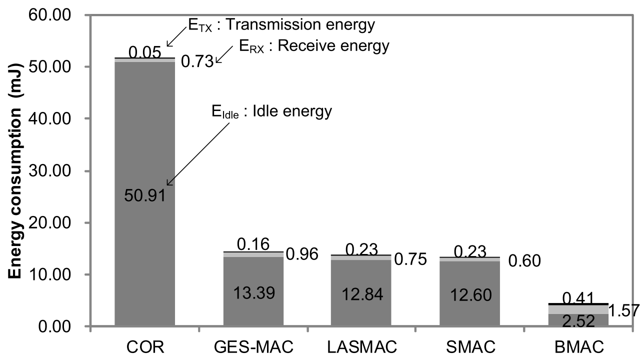


Fig. 21 Detailed energy consumption in a radio module.

4.3.2 Energy Consumption in a Radio Module

We measured the average energy consumed by a node's radio module of Tip30 and the result about the energy consumption of GES-MAC discussed in Sect. 4.2.1 is also measured similarly in the real sensor nodes. In order to measure the energy consumption of Tip30, we check the amount of time in idle, transmitting, receiving state, and calculate the energy consumption with the parameter in Table 2. One source periodically generates packets every ten seconds. COR has no duty cycle, BMAC has 1:100 duty cycle, and the other protocols have 1:4 duty cycle, as mentioned in Sect. 4.1. The result shown in Fig. 21 indicates that the energy consumptions of the duty cycle-based protocols are much lower than that of COR because the dominant part of energy consumption is idle energy. The energy consumption of GES-MAC is 28% of COR's energy. GES-MAC consumes 5% more energy than LASMAC on average because the test environment is less dense than the simulation environment (in simulation, 15% difference). The more consumed energy than LASMAC is cost for providing high goodput. Like simulation, BMAC is very energy efficient although it has difficulties in providing high goodput.

4.3.3 Discussion about the Overhead of GES-MAC

We need to discuss the overhead of GES-MAC to provide energy efficiency while maintaining high goodput. First, we discuss the number of control packets. We measured the number of control packets with Tip30 nodes. Figure 22 shows the number of detected control packets at a node in the center area. The control packets are RTS/CTS/ACK/SYNC/Preamble. Compare to COR and BMAC, GES-MAC needs more control packets. As GES-MAC uses synchronized duty cycle like LASMAC and SMAC, at the beginning of every listen periods, SYNC packets are shared even though there are no data packets to forward. We think that the control packet overhead is a cost of GES-MAC in order to balance high goodput and energy efficiency. Additionally, because BMAC needs a preamble packet for data forwarding, BMAC needs more control packets than COR. Interesting feature is that COR needs the

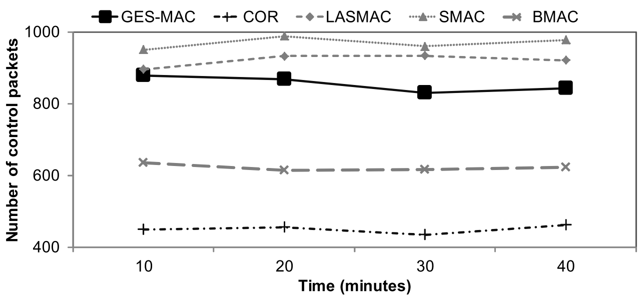


Fig. 22 Control packet overhead of each protocol.

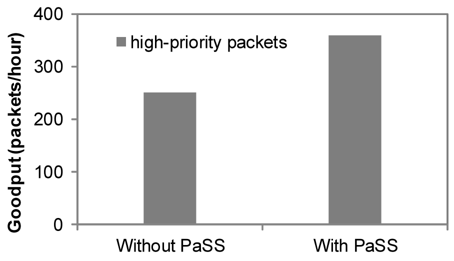


Fig. 23 Service differentiation in case of two close sources.

fewest number of control packets, but consumes very large energy.

After analyzing the compiled size of code, we thought that the code complexity is not proportional to the size of complied code because the size of code can be changed by programmer's skill. For example, in our version, the code size of SMAC is 50 kbytes and that of LASMAC is 46 kbytes. However, theoretically, LASMAC is more complex than SMAC because LASMAC needs node scheduling. Thus, with the code that we have, we cannot address the code complexity. Just note that the code size of GES-MAC is 56 kbytes.

Note that we set the size of sub-slot in Sect. 3.2.3 to two time slots (i.e. 2 ms). The computing power of Tip30 is not so good and each sensor has different computing power. Thus, if we set the sub-slot to 1 ms, two nodes try to send ACK simultaneously and collisions are induced. However, after changing the size of sub-slot to 2 ms, GES-MAC operates as we expected.

4.3.4 Experimental Results for Service Differentiation

We measured the service differentiability of GES-MAC, which is the effects of the Priority-aware Schedule Switching (PaSS) on the goodput of high-priority packets, with two close Tip30 sources. One source is a high-priority source (source1) and the other is a low-priority one (source2). In this test, the previous service differentiation using different Contention Window (CW) is applied. Source1 has 16 slots for CW and Source2 has 32 slots for CW. The time deadline is 1 second. In Fig. 23, without PaSS, if a source2 schedules SCs through the routing path earlier, the packet from source1 waits for long time gap and the packet does not reach the destination within one second. Thus, 250 high-priority packets meet the deadline. However, if PaSS is applied, even though SCs are already scheduled for source2, the high-priority packets are forwarded earlier by schedule switching. As a result, 360 high-priority packets (i.e. all populated packets) meet the deadline. The increased high-priority goodput means the good service differentiability of PaSS.

5. Conclusion

The goals of our work are to provide high goodput in duty-cycled environments and to provide service differentiation while maintaining high goodput. We proposed a method, called GES-MAC, which satisfies the goals. GES-MAC reduces idle listening energy consumption by using a synchronized duty cycle. *Cluster-based multi-hop scheduling* provides high goodput in duty cycle environments. *Priority-aware schedule switching* makes more high-priority packets reach the sink node. Experiments showed that GES-MAC reduced the idle listening energy consumption by about 70% by using a duty cycle. The goodput of GES-MAC is not much less than that of COR. GES-MAC also provides service differentiation with little overhead. Therefore, GES-MAC achieves our goals.

References

[1] W. Ye, J. Heigemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," Proc. Twenty-First Annual Joint Conf. of the IEEE Computer and Communications (INFOCOM 2002), pp.1567–1576, 2002.

[2] T. van Dam and K. Langendoen, "An adaptive energy efficient MAC protocol for wireless sensor networks," Proc. First ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys 2003), pp.171–180, 2003.

[3] J. Kim and D. Park, "An energy-efficient scheduling MAC protocol for wireless sensor networks," Proc. Fourth Annual IEEE Consumer Communications and Networking Conf, (CCNC 2007), pp.655–659, 2007.

[4] J. Kim and K. Park, "An energy-efficient Transport-controlled MAC protocol for wireless sensor networks," Elsevier Computer Networks, vol.53, pp.1879–1902, 2009.

[5] S. Du, A.K. Saha, and D.B. Johnson, "RMAC: A routing-enhanded duty-cycle MAC protocol for wireless sensor networks," Proc. 26th Annual Joint Conf of the IEEE Computer and Communications (INFOCOM 2007), pp.1478–1486, 2007.

[6] K. Kweon, H. Lee, and H. Yoon, "A retransmission-enhanced duty-cycle MAC protocol based on the channel quality for wireless sensor networks," IEICE Trans. Commun., vol.E93-B, no.11, pp.3156–3160, Nov. 2010.

[7] K. Nguyen, T. Nguyen, C.K. Chaing, and M. Motani, "A prioritized MAC protocol for multihop, event-driven wireless sensor networks," Proc. First Int'l Conf. on Mobile Adhoc and Sensor Systems Conference on Electrical Engineering (ICEE 2007), pp.47–52, 2007.

[8] A. Firoze, L. Ju, and L. Kwong, "PR-MAC a priority reservation MAC protocol for wireless sensor networks," Proc. Int'l Conf. on Communications and Electronics (ICCE 2006), pp.1–6, 2006.

[9] Z. Liu and I. Elhanany, "RL-MAC: A QoS-aware reinforcement learning based MAC protocol for wireless sensor networks," Proc. 2006 IEEE Int'l Conf. on Networking, Sensing and Control (ICNSC 2006), pp.768–773, 2006.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access wireless sensor networks," Proc. Second ACM Int'l Conf. on

Embedded Networked Sensor Systems (SenSys 2004), 2004.

[11] M. Buettner, G.V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," Proc. Fouth ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys 2006), 2006.

[12] Y. Sun, S. Du, O. Gurewitz, and D. Johnson, "DW-MAC: A low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," Proc. 9th ACM Int'l Symposium on Mobile Ad hoc Networking and Computing (MobiHoc 2008), pp.53–62, 2008.

[13] M.A. Yigitel, O.D. Incel, and C. Ersoy, "QoS-aware MAC protocols for wireless sensor networks: A survey," Comput. Netw., vol.55, pp.1982–2004, 2011.

[14] R EI-Azouzi, E. Sabir, SK Samanta, and R. EI-Khoury, "Asymptotic delay analysis and timeout-based admission control for ad hoc wireless networks with asymmetric users," Comput. Commun., vol.33, pp.2057–2069, 2010.

[15] K. Nguyen, U. Meis, and Y. Ji, "MAC^2: A multi-hop adaptive MAC protocol with packet concatenation for wireless sensor networks," IEICE Trans. Inf. Syst., vol.E95-D, no.2, pp.480–489, Feb. 2012.

[16] D. Kim, J. Kim, and K.H. Park, "An event-aware MAC scheduling for energy efficient aggregation in wireless sensor networks," Elsevier Computer Networks, vol.55, pp.225–240, 2010.

[17] M. Kurth, A. Zubow, and J.-P. Redlich, "Cooperative opportunistic routing using transmit diversity in wireless mesh networks," Proc. 27th Annual Joint Conf. of the IEEE Computer and Communications (INFOCOM 2008), pp.1310–1318, 2008.

[18] S. Biswas and R. Morris, "ExOR: Opportunistic multi-hop routing for wireless networks," Proc. 2005 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2005), pp.133–144, 2005.

[19] E. Felemban, C.G. Lee, and E. Ekici, "MMSPEED: Multipath multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks," IEEE Trans. Mobile Comput., vol.5, pp.738–754, 2006.

[20] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," Proc. 4th Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2004), pp.517–526, 2004.

[21] T.S. Rappapport, Wireless Communications: Principles and Practice, Prentice Hall, 1996.

[22] H. Zhang, L. Sang, and A. Arora, "Comparison of data-driven link estimation methods in low-power wireless networks," IEEE Trans. Mobile Comput., vol.9, pp.1634–1648, 2010.

[23] H. Zhang, A. Arora, and P. Sinha, "Learn on the fly: Data-driven link estimation and routing in sensor network backbones," Proc. 25th Annual Joint Conf. of the IEEE Computer and Communications (INFOCOM 2006), pp.1–12, 2006.

[24] A. Cerpa, J. Wong, M. Potkonjak, and D. Estrin, "Temporal properties of low power wireless links: Modeling and implications on multi-hop routing," Proc. 6th ACM Int'l Symposium on Mobile Ad hoc Networking and Computing (MobiHoc 2005), pp.414–425, 2005.

[25] I.F. Akyildiz, T. Melodia, and K.R. Chowdhury, "A survey on wireless multimedia sensor networks," Elsevier Computer Networks, vol.51, pp.921–960, 2007.

[26] K. Zeng, K. Ren, W. Lou, and P.J. Moran, "Energy-aware geographic routing in lossy wireless sensor networks with environmental energy supply," Proc. 3rd Int'l Conf. on Quality of service in heterogeneous wired/wirelss networks (QShine 2006), 2006.

[27] B. Karp and H.T. Kung, "Greedy perimeter stateless routing for wireless networks," Proc. 6th Annual Int'l Conf. on Mobile Computing and Networking (MobiCom 2000), pp.243–254, 2000.

[28] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy efficient forwarding strategies for geographic routing," Proc. 2nd Int'l Conf. on Embedded Networked Sensor Systems (SenSys 2004), pp.108–121, 2004.

[29] H.A.B.F. Oliveira, A. Boukerche, E.F. Nakamura, and A.A.F. Loureiro, "An efficient directed localization recursion protocol for wireless sensor networks," IEEE Trans. Comput., vol.58, no.5, pp.677–691, 2009.

[30] Maxfor TIP30 Specifications. "http://www.maxfor.co.kr"

SangKwon Moon received the B.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2002 and the M.S. degree in electrical engineering from KAIST in 2004. He is currently a Ph.D. student at KAIST. His research interests include sensor network, ad-hoc networks, energy efficiency, multi-channel network, ubiquitous computing and middleware.

Jong-Woon Yoo received the B.S., M.S., and Ph.D. degrees in electronic engineering from KAIST, Korea in 2005, 2007, and 2011, respectively. Since Feb. 2011, he has been in a postdoctoral position in Computer Engineering Research Laboratory (CORELAB) at KAIST, working on energy-efficient mobile networking, in-vehicle distributed embedded systems, and wireless sensor networks. His research interests also include human-computer interaction (HCI) in ubiquitous computing, and manycore computing.

Jaesub Kim received the B.S. degree in electrical engineering from Kyungpook National University in 2000, the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2002, and the Ph.D. degree in electrical engineering from KAIST in 2009. His research interests include sensor network, internet server systems, and operating systems.

Kyu-Ho Park received the B.S. degree in electronics engineering from Seoul National University, Korea in 1973, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 1975, and the DrIng degree in electrical engineering from the University de Paris XI, France in 1983. He has been a Professor of the Division of EECS, KAIST since 1983. He was a president of Korea Institute of Next Generation Computing in 2005–2006. His research interests include computer architectures, file systems, storage systems, ubiquitous computing, and parallel processing. He is a member of KISS, KITE, Korea Institute of Next Generation Computing, IEEE and ACM.