Ann. Telecommun. (2011) 66:567–582
DOI 10.1007/s12243-010-0226-7

nW-MAC: multiple wake-up provisioning in asynchronously scheduled duty cycle MAC protocol for wireless sensor networks

Md. Obaidur Rahman · Muhammad Mahbub Alam · Muhammad Mostafa Monowar · Choong Seon Hong · Sungwon Lee

Received: 5 January 2010 / Accepted: 5 November 2010 / Published online: 25 November 2010

Abstract To reduce the energy cost of wireless sensor networks (WSNs), the duty cycle (i.e., periodic wake-up and sleep) concept has been used in several medium access control (MAC) protocols. Although these protocols are energy efficient, they are primarily designed for low-traffic environments and therefore sacrifice delay in order to maximize energy conservation. However, many applications having both low and high traffic demand a duty cycle MAC that is able to achieve better energy utilization with minimum energy loss ensuring delay optimization for timely and effective actions. In this paper, *n*W-MAC is proposed; this is an asynchronously scheduled and multiple wake-up provisioned duty cycle MAC protocol for WSNs. The *n*W-MAC employs an asynchronous rendezvous schedule selection technique to provision a maximum of *n* wake-ups in the operational cycle of a receiver. The proposed MAC is suitable to perform in both low- and high-traffic applications using a reception window-based medium access with a specific RxOp. Furthermore, per cycle multiple wake-up concept ensures optimum energy consumption and delay maintaining a higher throughput, as compare to existing mechanisms. Through analysis and simulations, we have quantified the energy-delay performance and obtained results that expose the effectiveness of *n*W-MAC.

Keywords Wireless sensor network (WSN) · Medium access control (MAC) · Wake-up schedule · Operational cycle · Duty cycle · Energy · Delay

1 Introduction

Currently, one of the main challenges for wireless sensor networks (WSNs) is the handling of large variations in traffic loads (i.e., low and high) for various applications. Additionally, many monitoring applications (e.g., fire alarm, intruder detection, tracking, etc.) demand timely and effective actions with minimum delay. However, the energy-efficiency constraint often makes it challenging for WSNs to perform in real-time; in such applications a very low periodic traffic primarily exists for a longer period of time in the absence of event traffic. Therefore, the WSN design protocols need to consider an energy-delay trade-off in order to fulfill such application requirements.

Imperative and co-related research have shown energy as the most decisive resource for any WSN, including monitoring networks; the demand for extended network life time motivates the design of energy efficient medium access control (MAC) protocols [1, 2]. Consequently, the periodic wake-up and sleep schedule (namely *duty-cycle*)-based MAC is introduced to reduce

Md. O. Rahman · M. M. Alam · M. M. Monowar · C. S. Hong (✉) · S. Lee
Department of Computer Engineering, College of Electronics and Information, Kyung Hee University (Global Campus), Seoul, South Korea
e-mail: cshong@khu.ac.kr

Md. O. Rahman
e-mail: mdobaidurrahman@gmail.com

M. M. Alam
e-mail: mahbub@networking.khu.ac.kr

M. M. Monowar
e-mail: monowar@ieee.org

S. Lee
e-mail: drsungwon@khu.ac.kr

energy consumption [1–4]. A sensor node in these protocols has its own wake-up schedule, i.e., maintains a wake-up cycle (hereafter, we interchangeably refer to this as either an *operational cycle* or a *cycle*). The cycle duration is more adaptable for existing duty cycle MAC protocols in working energy efficiently in low traffic only. Whereas, the blow of the of event data or heavy traffic is still unrevealed in designing such protocols.

To perform in real-time it is important to ensure a minimum level of delay. Unfortunately however, conventional sensor MAC design places the emphasis on maximizing energy conservation [1, 2]. In typical protocols, each node wakes-up once per cycle to receive any potential data and sleeps for the remainder of the cycle period. Senders that cannot transmit their data in that wake-up time have to wait for at least a receiver's sleep period; this is defined as a sleep delay in D-MAC [5]. Thus, in a multi-hop path, the worst-possible delay is bounded by either the cycle or sleep periods of the intermediate nodes. Moreover, we argue that the per cycle single wake-up strategy for the receiver is not suitable for event traffic; when an event occurs the delay increases if the nodes cannot forward the event-triggered heavy traffic due to the sleep periods. Although the concept of multiple packet reception at each wake-up is introduced in [1, 3], in heavy traffic environment their unbounded reception endures more collisions resulting in very low throughput and large delay due to retransmissions.

So far, based on rendezvous selection between sender and receiver, duty cycle MAC protocols are mainly categorized into *synchronous* and *asynchronous* protocols. In a synchronous MAC [4, 6], nodes in the same neighborhood are time-synchronized for simultaneous wake-up at the beginning of a common cycle and transmit their data using the basic carrier sense multiple access with collision avoidance (CSMA/CA). However, nodes simultaneous wake-up in synchronous protocols results in higher contentions and collisions, and the clock synchronization overhead makes it difficult to implement in a multi-hop WSN.

Conversely, in asynchronous protocols, nodes maintain their individual cycles at random and the sender-receiver rendezvous is selected either through the sender or receiver initiated medium access. In sender initiated MAC [2, 3], a long preamble (equivalent to at least a receiver's cycle or sleep period) from the sender is used to rendezvous with the receiver. Such a preamble causes a significant amount of energy waste and delay. In comparison to its sender initiated counterparts, a receiver initiated MAC [1] provides better energy and delay performance, by avoiding the unnecessary medium occupancy time for a preamble transmission. However, the existing receiver initiated schemes still suffers due to idle listening [4] and sleep delay at the sender end. Furthermore, recent asynchronous MAC protocols [1, 3] lags with traffic variations and rarely handle heavy traffic scenarios. Hence, improvements are still needed for the issues discussed.

In this paper, motivated by the above observations, *n*W-MAC is proposed. This is an asynchronously scheduled receiver- initiated duty cycle MAC protocol for WSNs comprising *n* wake-up provisions in the receiver cycle. This protocol attempts to satisfy the following:

– Minimize energy consumption with optimal delay under low traffic conditions.
– Maximize energy utilization with improved throughput and minimum delay in the handling of heavy traffic.

The major contributions of *n*W-MAC can be summarized as follows: (1) An innovative multiple wake-up provisioning in sensor MAC is introduced, utilizing an energy and delay efficient asynchronous rendezvous schedule selection by the receiver. (2) A reception window-based bounded channel access is used, limiting the reception opportunity (RxOp) of a receiver in receiving multiple packets at each wake-up. (3) Adaptability is ensured on-demand basis in using the *n* wake-up provisions by the receiver at every cycle. (4) Performance of the *n*W-MAC is analyzed in detail and evaluated through extensive simulations.

The rest of the paper is organized as follows: Section 2 presents a background study on existing sensor MAC protocols, Section 3 introduces the preliminary design considerations, Section 4 includes detailed protocol operations, Section 5 analyzes *n*W-MAC in terms of energy consumption and delay, Section 6 presents performance evaluations by simulation efforts and comparisons, and Section 7 provides our conclusions with future considerations.

2 Related work

In recent years, a number of MAC protocols have been proposed for WSN, and as mentioned in Section 1, existing asynchronous MAC protocols can be categorized into sender- and receiver-initiated MAC. The very early sender- initiated MAC protocol, B-MAC [2], optimizes the WSN energy performance by employing a preamble sampling technique. In recent research, [1, 3, 7], it has been determined that the detection of a preamble forces a node to receive the entire preamble causing large overhearing [4] oriented energy loss at the non-intended receivers. Although B-MAC is an energy-efficient protocol and works well in very low traffic, it is unable to avoid preamble overhead and sleep delay.

The X-MAC [3] overcomes the overhearing problem with the long preamble transmission in B-MAC. In X-MAC, strobe preambles (having receiver IDs) are transmitted by the sender, as shown in Fig. 1a. At the asynchronous wake-up (i.e., after the wake-up interval), if the receiver detects the strobe preamble(s), it simply sends an acknowledgment (ACK), otherwise it goes to sleep. Upon receiving an early ACK from the receiver, the sender then transmits the data and receives the data ACK; hence, X-MAC partially reduces the preamble transmission time. However, we have identified that X-MAC still has two basic problems. Firstly, it wastes energy due to the strobe preambles since, on average for half of the receiver's cycle (i.e., wake-up interval), a sender needs to transmit the preambles to be in rendezvous with the receiver, as shown in Fig. 1a. Secondly, at the sender end, the sleep delay due to the receiver's wake-up interval (or cycle) is very large; on average, this delay is half of the cycle period at each hop. Additionally, X-MAC does not consider the consequences of high traffic, which is the primary concern in the handling of a monitoring network.

The asynchronous protocol WiseMAC [7] is limited to work only for the downlinks of infrastructure WSNs, and adaptively minimizes the preamble length. At each reception, the receiver informs the sender about the next wake-up schedule. Senders store the receiver's wake-up time (offset) and accordingly reduce the preamble length for further transmissions for that

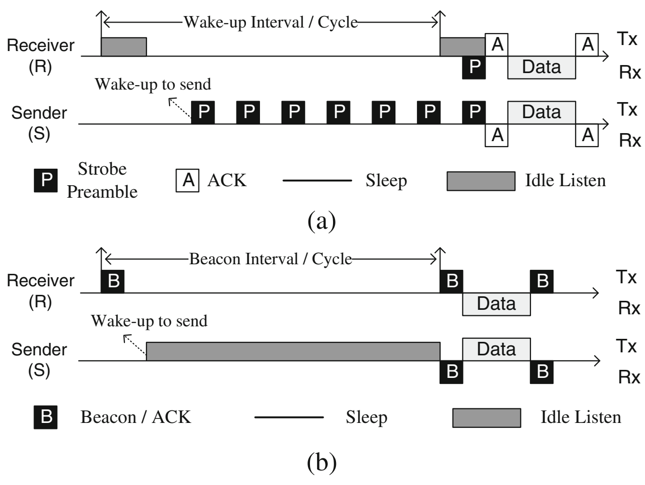


Fig. 1 Protocol operations: **a** X-MAC and **b** RI-MAC

specified wake-up. *n*W-MAC uses the offset storing specification of WiseMAC where each sender-receiver pair asynchronously selects their rendezvous.

The most recent protocol, RI-MAC [1], brings the first receiver initiated MAC concept into WSNs. The operation of RI-MAC is shown in Fig. 1b; a receiver at each wake-up cycle (or beacon interval) sends a request message (i.e., beacon) for potential data reception. Thus, the receiver initiated concept is able to decrease the energy loss and medium occupancy time associated with a preamble as well as the delay that occurs in the sender initiated protocols [2, 3, 7]. A sender in RI-MAC listens and checks the medium for a request beacon from the receiver. Upon its receipt, the sender transmits the data and receives the data ACK (i.e., also a beacon for more data request). Although with low traffic RI-MAC performs efficiently, the medium access mechanism of this protocol has a number of shortcomings. We have observed that in RI-MAC the idle listening and sleep delay period at the sender end are very long due to the receiver's beacon interval (as in Fig. 1b). On average, the idle listening and sleep delay are lengthened to half of the receiver's cycle at each hop. Therefore, RI-MAC still causes unnecessary energy loss and delay. Furthermore, with heavy traffic the unbounded multiple packets reception at each wake-up of the receiver increases the overall network contentions. Hence, collision oriented retransmissions in RI-MAC minimize throughput causing significant delay in multi-hop path.

Over the years, several synchronous MAC protocols, (e.g., S-MAC [4], D-MAC [5], T-MAC [6], R-MAC [8], DW-MAC [9]) have been proposed for WSNs. The energy-efficient S-MAC [4] is the first synchronous MAC protocol for WSNs, and several extensions of this innovative work are introduced later on. D-MAC [5] is such a protocol with a precisely synchronized staggered solution to achieve a minimum delay over a multi-hop path, where nodes first receive a packet and immediately transmit it to downstream. R-MAC [8] provides another staggered solution using a control packet (*pion frame*). In *n*W-MAC, we also propose a similar but asynchronously scheduled staggered solution, where network wide synchronization is avoided and unlike [5, 8], receivers can receive multiple packets at each wake-up

There exists a few well-known hybrid MAC protocols such as SCP-MAC [10] and Funneling-MAC [11]. These protocols use the advantageous properties of both the synchronous and asynchronous schemes. In SCP-MAC, nodes are precisely synchronized like S-MAC, whereas in Funneling-MAC, only the nodes near the sink are synchronized.

Despite the energy and delay efficiency, existing synchronous and hybrid MAC protocols are not good choices for WSNs due to the synchronization overhead and their implementation complexity. *n*W-MAC uses a determined asynchronously scheduled duty cycle approach; each receiver maintains an individual cycle at random and senders follow the time differences to predict the scheduled rendezvous with the receiver considering the potential clock drifts.

3 System model and preliminaries

3.1 Network model

*n*W-MAC is designed for a typical WSN scenario, as shown in Fig. 2a; it comprises multiple unidirectional data flow converging toward a single point or sink, resulting in a tree topology. The research for D-MAC [5] justifies our assumption of the tree topology for a duty cycle MAC protocol; the network consists of stationary sensor nodes with a single routing path from each node to the sink.

*n*W-MAC is a receiver initiated MAC, there are N receiving nodes in the network, and the ith receiver is denoted as r_i. Each receiver has one downstream node and multiple upstream nodes. The downstream node of r_i is denoted as d_i and the jth upstream node of r_i is denoted as u_{ij}.

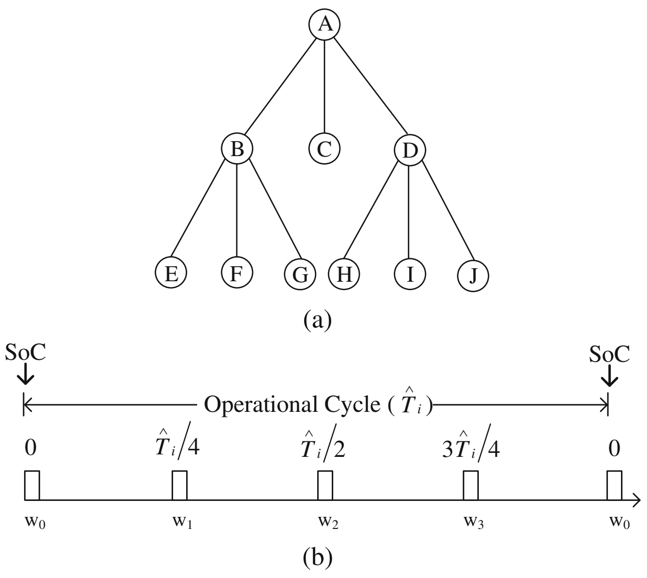


Fig. 2 **a** Network model and **b** in operational cycle ($\widehat{T}_i$) of node r_i, the provisions of n wake-ups $\{w_0, w_1, w_2, \cdots, w_{(n-1)}\}$; here $n = 4$

A flow is defined as the traffic from a particular source node (either periodically, or upon the detection of any event, or a combination of both). *n*W-MAC considers the unicast transmission of a flow.

3.2 Assumptions

We assume that each receiver has an operational cycle and each cycle has an active operational period which determines the duty cycle of a node [1, 3, 4]. However, the operational cycle and duty cycle for *n*W-MAC are defined as follows:

Definition 1 *Operational Cycle*: an operational cycle or a node cycle is the time interval that includes multiple wake-up provisions. Let $\widehat{T}_i$ denote the cycle of r_i.

Definition 2 *Duty Cycle*: the duty cycle of a node is defined as the ratio between a node's active time to its entire cycle time. Active time includes all of the actions and activities of a node (i.e., channel access at single or multiple wake-ups, medium listening, transmission and reception etc.).

In a cycle, *n*W-MAC provisions a number of wake-ups for a receiving node. Let n denote the number of wake-ups in a cycle, and w_k denote the identification (ID) of the kth wake-up where $\{k = 0, 1, \cdots, (n-1)\}$, as shown in Fig. 2b. The value of n is set equal to 4 based on extensive simulations; it produces the optimum energy-delay trade-off for a wide variation of simulation environments. All of the subsequent sections of this paper use this same value for n. We further assume that each data transmission uses a request-to-receive (RTR) packet initiated by the receiving node at each wake-up. RTR is similar to the beacon concept in RI-MAC [1], however, besides a request for data and acknowledgment a successful reception, a receiver in *n*W-MAC also uses it for wake-up scheduling.

Definition 3 *Wake-up Offset*: The n wake-up offsets of a receiver originate from the start-of-cycle (SoC), t_{SoC} and the offset of the w_k wake-up of r_i is obtained by

$$t^i_{w_k} = t_{\text{SoC}} + (k \mod n)\frac{\widehat{T}_i}{n}, \tag{1}$$

Therefore, in a cycle, the interval between consecutive wake-up provisions of r_i is $\frac{\widehat{T}_i}{n}$.

3.3 Initialization

We assume that each node chooses a random cycle duration at startup and remains active for a number of cycles. Furthermore, each node announces its cycle length to the upstream nodes using the basic CSMA/CA mechanism. However, the active period of an upstream node u_{ij} initially depends on the cycle length of the receiver r_i which u_{ij} extracts from the announcement of $\widehat{T}_i$. The active window of u_{ij}, denoted as T_{active}, is given by

$$T_{\text{active}} = \frac{\widehat{T}_i}{n} + T_r, \tag{2}$$

where, T_r is referred to as the RTR window which is defined as the duration of the transmission of the RTR packet and the receipt of its acknowledgment. During initialization, node u_{ij} asynchronously wakes-up at random; after this point it wakes-up at a fixed interval of $\widehat{T}_i$ (i.e., the cycle length of u_{ij}'s receiver r_i). After wake-up, u_{ij} checks the medium for a period of T_{active} before going back to sleep. Therefore, if r_i transmits an RTR packet in each of its n wake-ups, it is guaranteed that u_{ij} receives one of the n RTR packets of r_i. This assumption can be verified by the dominating-awake-interval technique [12] and overlapping principle [13].

4 *n*W-MAC protocol operation

4.1 nW-MAC overview

In *n*W-MAC, an asynchronous scheme derives maximum n wake-up schedules per cycle for each receiver. The schedule selection is mainly designed to optimize the energy consumption and delay in accessing the medium.

After initialization and schedule selection, nodes follow the wake-up and sleep schedules. In basic *n*W-MAC, a receiver wakes-up at a scheduled reception time and sends an RTR in order to receive data from the senders. In contrast, a sender only wakes-up prior to the receiver's specified wake-up time, and sends data upon receiving the RTR. Furthermore, to receive multiple packets at each wake-up, a RxOp limit is used for the receiver.

To fulfill the demand of an event monitoring network, an adaptive version of *n*W-MAC is also proposed; under very low periodic traffic, a receiver maintains a per cycle single wake-up-based asynchronous staggered schedule (see Section 4.4.1). Conversely, in response to event-triggered heavy traffic, each receiver makes an on-demand adaptive and additive use of the provisions for n wake-ups in every cycle.

4.2 Asynchronous schedule selection

The proposed asynchronous scheme first selects a rendezvous schedule for each sender-receiver pair at one of the n wake-ups of the receiver. To maximize energy conservation, both sender and receiver store their selected rendezvous, and accordingly, they wake-up to send and receive, respectively.

To select the schedule, a receiver r_i wakes-up at w_0, w_1, w_2, and w_3 within the cycle ($\widehat{T}_i$). At each wake-up, it transmits an RTR packet to its upstream node(s) u_{ij} if the medium is found free. Otherwise, the receiver waits for an RTR contention window period, denoted as CW_{RTR}, and transmits the RTR afterward. On the contrary, in every $\widehat{T}_i$ interval, all u_{ij} asynchronously listen to the medium for a period T_{active} (derived in Section 3.3). Therefore, if there is no collision originated loss, it can be guaranteed that any of the RTR packets (out of 4) transmitted by r_i at different w_ks, is received by the upstream nodes [12, 13]. Upon receiving the RTR, the upstream nodes wait for a *SIFS* and contend with a random back-off (0 to CW_{RTR}). An u_{ij} with an early back-off expiration replies with an acknowledgment (ACK) and stores that w_k wake-up offset of r_i as its scheduled transmission (Tx) rendezvous in every $\widehat{T}_i$.

After receipt of the first ACK from any u_{ij}, r_i waits for a maximum back-off (CW_{RTR}) to receive further ACK from another node, if any. The upstream node(s) that looses the previous contention pauses the back-off counter; when the medium becomes free it resumes counting to send an ACK. Accordingly, the w_k wake-ups, at which r_i receives an ACK from its upstream nodes, are selected as the scheduled reception (Rx) rendezvous; therefore, for r_i the maximum number of scheduled wake-ups during $\widehat{T}_i$ are less than or equal to n.

As in Fig. 3, node A transmits an RTR at each of its four wake-ups (w_0, w_1, w_2, and w_3), for every cycle; at w_1 one of the RTRs is received and acknowledged by

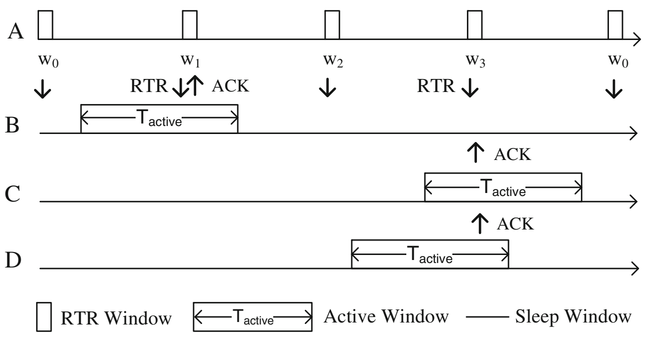


Fig. 3 Duty cycle after initialization. Node A's upstream nodes B, C and D (as in Fig. 2a) have the active time T_{active} in every $\widehat{T}_A$ interval

node B. Therefore, w_1 is selected as the Tx rendezvous for B with the receiver A. Again, nodes C and D receive the RTR at w_3 of A. Now, if D sends an ACK first, C pauses the back-off and resumes whenever node D finishes transmission and vice versa. Hence, w_3 is selected as the Tx rendezvous for C and D with their receiver A. Finally, in subsequent cycles node A wakes-up only at w_1 and w_3, since these are the selected Rx rendezvous for A.

An upstream node further selects a start-of-cycle (t_{SoC}) for its own w_0 wake-up. Let an u_{ij} have a Tx rendezvous at $t^i_{w_k}$ (i.e., w_k wake-up of r_i). Thus, the t_{SoC} for u_{ij} is chosen as

$$t_1 = t^i_{w_k} - \frac{\widehat{T}_i}{2n}, \quad t_2 = t^i_{w_k} - (g_1 + g_2),$$
$$t_{\mathrm{SoC}} = rand\,(t_1, t_2), \tag{3}$$

where $rand(t_1, t_2)$ is a function that generates a random start-of-cycle ranging from the time offset t_1 to t_2. Hence, for $n = 4$, the t_{SoC} or w_0 wake-up of u_{ij} can be, at most, a $\frac{\widehat{T}_i}{8}$ unit time earlier than the w_k wake-up of r_i. Moreover, the guard time g_1 is considered for the potential clock drift [7], and the guard time g_2 provides the opportunity for receiving at least one data packet (from u_{ij}'s upstream node) by u_{ij} at its w_0.

$$g_2 = (\mathrm{slotTime} \times \mathrm{CW_{RTR}}) + T_{\mathrm{RTR}} + \mathrm{SIFS} + (\mathrm{slotTime} \times \mathrm{CW_{Data}}) + T_{\mathrm{Data}} + \mathrm{SIFS} + T_{\mathrm{ACK}}, \tag{4}$$

here, T_{RTR}, T_{Data}, T_{ACK} are the required time periods for transmitting RTR, data, and ACK packets, respectively. The $\mathrm{CW_{Data}}$ is the contention window for the data packet, and slotTime is used to calculate the maximum back-off times.

Considering Fig. 4, suppose that node B has Tx rendezvous at w_1 of A; hence, t_{SoC} of B is selected between $\frac{\widehat{T}_A}{8}$ and $(\frac{\widehat{T}_A}{4} - (g_1 + g_2))$. After selecting t_{SoC}, node B commences sending an RTR at each of its $w_0, \cdots, w_3$ wake-ups.

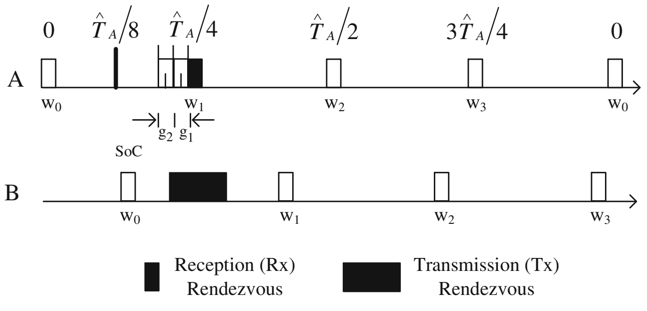


Fig. 4 Start-of-Cycle (*SoC*) selection for an upstream node of A; here, upstream node B has the Tx rendezvous at w_1 wake-up of A

The proposed scheduling is adaptive and scalable to any topology changes in a WSN. To join the network, a new node or node having link failures listens to the medium for at least a maximum cycle period, $\widehat{T}_{\max}$, a system parameter. Therefore, within the coverage area, that node is guaranteed to hear RTR(s) from its neighbor(s) and is then able to select its receiver and schedules during regular MAC operations.

4.3 Basic nW-MAC

In basic nW-MAC, a receiver follows a reception window-based MAC, where a *reception window* is defined as

Definition 4 *Reception Window*: A variable length active period, T^i_{rw}, which reflects the RxOp limit of the receiver r_i in a specific reception rendezvous.

In a scheduled Rx rendezvous at w_k (or $t^i_{w_k}$), node r_i's maximum T^i_{rw} length is bounded by either of two other offsets: (1) r_i's own Tx rendezvous at a downstream node d_i's wake-up offset $t^{d_i}_{w_k}$, and (2) r_i's another Rx rendezvous at offset $t^i_{w_l}$, where $l \neq k$. Thus, receiver r_i calculates T^i_{rw} as

$$T_1 = t^i_{w_l} - t_{\mathrm{current}}, \quad T_2 = t^{d_i}_{w_k} - t_{\mathrm{current}},$$
$$T^i_{\mathrm{rw}} = \min(T_1, T_2) - g_1, \tag{5}$$

where t_{current} is the current time and the guard time, g_1, is excluded for potential clock drift. Receiver r_i continuously update the instantaneous T^i_{rw}. As shown in Fig. 5, in the cycle $\widehat{T}_B$, node B has a scheduled Rx

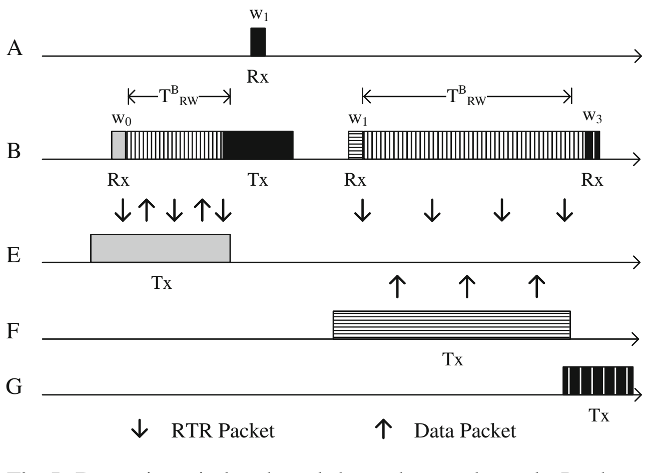


Fig. 5 Reception window-based channel access by node B, where B has the downstream node A and upstream nodes E, F, G

rendezvous at w_0, w_1 and w_3 for the upstream nodes E, F and G, respectively. Here, the maximum T^B_{rw} length at w_0 is bounded by B's Tx rendezvous with receiver A (at w_1 of A). The maximum T^B_{rw} at w_1 is limited by B's other Rx rendezvous (at w_3 of B).

Algorithm 1 represents the core operations of the proposed reception window-based basic nW-MAC. In every cycle, a receiver r_i periodically wakes-up at its scheduled Rx rendezvous and transmits an RTR (if the medium is idle). Prior to sending the first RTR at each wake-up, r_i performs a back-off (0 to $\mathrm{CW_{RTR}}$) to avoid collisions (line 3). However, if the medium is busy, r_i waits until the minimum of either the reception window (T^i_{rw}) or a duration of $\frac{\widehat{T}_i}{2n}$ (line 2). Meanwhile, it sends the RTR whenever the medium becomes idle. After sending the RTR, r_i waits for a *SIFS* plus a maximum back-off ($\mathrm{CW_{Data}}$) (line 5) in order to receive data from the intended sender(s). The expiration of this period forces the receiver to sleep (line 22) since it indicates no data or a busy medium at the sender end. Conversely, upon receiving any data packet within that time period (line 6), r_i waits for a SIFS, and transmits another RTR (line 8 or 12) as a data acknowledgment (i.e., the ACK bit is set to 1).

Algorithm 1 Reception Window Based nW-MAC

```
1.  for each receiving node $r_i \in N$ at $w_k$ do
2.    while (min($T^i_{rw}$, $\frac{\widehat{T}_i}{2n}$)) do
3.      if (Medium == IDLE && Back-off == FINISH) then
4.        Send(RTR); /* Data Request = 1, ACK = 0 */
5.        while ($SIFS + (slotTime \times CW_{Data})$) do
6.          if (Packet Received) then
7.            if ($T^i_{rw} > (SIFS + T_{RTR} + g_1)$) then
8.              Send(RTR); /* Data Request = 1, ACK = 1 */
9.              Sleep = FALSE;
10.             Goto line 5;
11.           else
12.             Send(RTR); /* Data Request=0, ACK=1 */
13.             Sleep=FALSE;
14.             if ($T_1 \leq T_2$) then
15.               Wake-up till $t^i_{w_l}$;
16.             else
17.               Wake-up till $t^{d_i}_{w_k}$;
18.             end if
19.           end if
20.         end if
21.       end while
22.       Sleep=TRUE;
23.     end if
24.   end while
25. end for
```

Moreover, a data request bit is set to 1 in the same RTR (line 8), when, in addition to receive a further data packet, T^i_{rw} is found greater than g_1 plus the time required to send the last RTR (i.e., the ACK) (lines 7-10). Hence, the reception window allows a receiver to receive additional, as well as multiple packets, at each wake-up. However, when T^i_{rw} goes below a specified duration (lines 11-18), the data request bit is set to 0 in the last RTR (line 12), and instead of sleeping, a receiver is awakened until its next scheduled Rx or Tx rendezvous (either till $t^i_{w_l}$ or $t^{d_i}_{w_k}$). This assumption considers a trade-off for saving the radio initiating energy [2].

In contrast, to adjust the clock drift a sender wakes-up g_1 earlier than its Tx rendezvous. The sender listens to the medium for a $T_r + 2g_1$ period to receive a potential RTR from the receiver, where T_r is the RTR window. However, due to the busy medium at the receiver end, sender(s) might not get an RTR within that duration and thereafter, waits for another $\frac{\widehat{T}_i}{2n}$ period. If there is still no RTR packet for the sender, it goes to sleep until the receiver's next cycle.

Upon receiving an RTR, the sender(s) contends with a random back-off (0 to $\mathrm{CW_{Data}}$). A node with an early back-off expiration sends a data packet and waits for an ACK from the receiver, while another sender(s) (if any) pauses the back-off and waits for more data request (in another RTR) from the receiver. The same or a different sender(s) further transmits data in the same way as if the receiver requests more data, otherwise it goes to sleep.

As shown in Fig. 5, at w_0 wake-up, receiver B sends the RTR packets until the reception window allows and receives multiple data packets from E in response to each RTR. As soon as the T^B_{rw} expires, B remains active until the w_1 wake-up of downstream node A.

In basic nW-MAC, the overall energy cost may increase as a result of the additional active period ($\frac{\widehat{T}_i}{2n}$) for both the receiver and sender(s). However, to avoid the sleep delay this energy trade-off is deemed necessary. Only in the worst-possible case does a receiver or sender have to lengthen the active period till $\frac{\widehat{T}_i}{2n}$. Note that for $n = 4$, this period is only one eighth of the receiver's cycle ($\widehat{T}_i$), and potentially happens only when the medium is highly loaded [14].

4.4 Adaptive nW-MAC

The unpredictable nature of monitoring networks leads to the mixing of high volume of event traffic with the mostly durable low periodic data. Existing sensor MAC protocols [1, 3] rarely handle the event traffic phenomena. Therefore, adaptive nW-MAC is designed to work with event data using the same reception window-based concept of Section 4.3.

4.4.1 Asynchronous staggered schedule

In the proposed asynchronous schedule selection (Section 4.2), there is the possibility that in every cycle a node might experience a fixed delay prior to forwarding any data to downstream. The consequences of such a delay at every hop may result in a larger end-to-end delay. As in Fig. 6a, node B has the Tx rendezvous at w_1 wake-up of receiver A, whereas, it has the Rx rendezvous for upstream nodes C and D at its w_0 and w_1 wake-ups, respectively. Thus, packets received by B at w_1, experience a fixed delay (greater than $\frac{\widehat{T}_A}{8}$), since it can only send the received packets at the w_1 wake-up of A. Similar delay occurs for the packets received by C at its w_1 and w_2 wake-ups, as C can only send at w_0 wake-up of B.

Moreover, with very low periodic traffic the use of multiple wake-ups in every cycle appears careless from an energy- efficiency point of view. Therefore, to avoid these problems *n*W-MAC proposes an adaptive staggered scheduling, which is able to maintain the optimum energy consumption and delay for nodes with different hops on a data gathering tree. To accomplish this during regular MAC operations (using RTRs), a receiver r_i announces a single Rx rendezvous at the $w_{k_{\min}}$ of a cycle, where $w_{k_{\min}}$ has a minimum forwarding delay toward the downstream node d_i.

In subsequent cycles, the senders adhere to the new rendezvous schedule with the receiver. Nodes on different hops adaptively build up an almost staggered data forwarding which we refer to as an *asynchronous staggered schedule*. In Fig. 6c, node B selects the new Rx rendezvous at w_0, since w_0 has a minimum delay with the w_1 wake-up of A; upstream nodes C and D follow the new Tx rendezvous at the w_0 wake-up of B. Similarly, node C chooses the Rx rendezvous at w_0 due to its minimum delay with the w_0 wake-up of receiver B.

Adaptation of this approach at the boundary nodes results in a single wake-up-based asynchronous staggered solution for each cycle, as shown in Fig. 6c. Analysis shows that for a chain topology, the asynchronous staggered schedule limits the per hop average delay within an approximate maximum bound of $\frac{\widehat{T}_i}{2n}$; for $n = 4$, this delay is only one eighth of the receiver's cycle ($\widehat{T}_i$).

4.4.2 Multiple wake-up schedule

An increase in the traffic load may occur due to event detection, variation in the data generation rate, etc. As mentioned, *n*W-MAC has the provisions for n wake-ups for each cycle of a receiver; whenever it is necessary receivers utilize these provisions. Note that the traffic load detection [14, 15] is out of the scope of this paper, rather, the *n*W-MAC protocol predicts and performs adaptively to deal with varying loads.

While operating in the single wake-up-based staggered schedule, a receiver r_i decides that another wake-up in the same cycle is necessary, based on the determination of full utilization of the reception window. More specifically, when the T^i_{rw} of a receiver r_i expires, due to the downstream node d_i's wake-up offset $t^{d_i}_{w_k}$, r_i presumes that senders still have data to send. However, due to the bounded reception window the receiver cannot receive the additional data.

A receiver adapts to the prediction of more data, and interjects an additional wake-up in the same cycle. To accomplish this, in addition to the $w_{k_{\min}}$ wake-up, r_i announces (using RTR) another Rx rendezvous at the next minimally delayed (i.e., in terms of forwarding

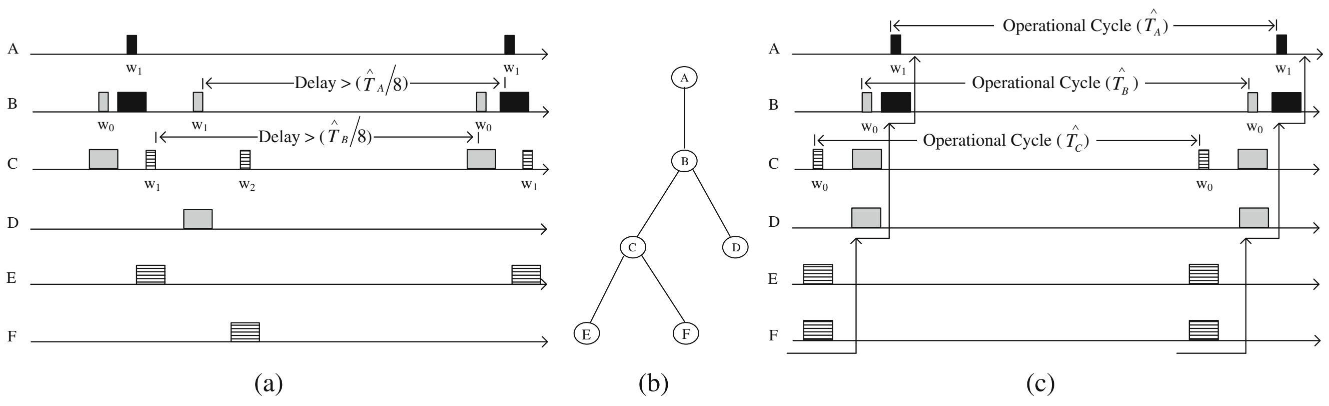

A
B
C
D
E
F
w1
w0
w2
Delay > ($\widehat{T}_A/8$)
Delay > ($\widehat{T}_B/8$)
Operational Cycle ($\widehat{T}_A$)
Operational Cycle ($\widehat{T}_B$)
Operational Cycle ($\widehat{T}_C$)
(a)
(b)
(c)

Fig. 6 **a** Rx and Tx Rendezvous schedules for the nodes of the tree in Fig. 6b, **b** A data gathering tree, and **c** Single wake-up-based asynchronous staggered schedule for the nodes of the tree in Fig. 6b toward d_i) wake-up of the same cycle. Senders with more data are informed (snoop the RTR) of the additional wake-up; hence, they immediately go to sleep until the receiver's new wake-up. Thus, by an on-demand basis, a receiver additively utilizes all of its n wake-ups when necessary, due to the full utilization of the reception window at any newly added wake-up. In contrast, under utilization of the reception window at any new wake-up, drives a receiver to deduct it from the wake-up list.

Therefore, a per cycle multiple wake-up strategy decreases contentions and collisions at the subsequent Rx rendezvous of a receiver. With high traffic, this strategy is able to ensure a higher throughput with minimum delay, which are the critical goals of nW-MAC. Although multiple wake-ups of a receiver require additional radio initiating energy [2], this additional energy would be worthwhile in order to minimize delay and maximize throughput.

5 Analysis of *n*W-MAC

To verify the performance of nW-MAC, several observations on the energy consumption and delay optimization are analyzed in this section. Throughout the analysis, the notations contained in Table 1 are used, where the power levels of CC2420 radio are considered in the energy measurements.

Table 1 Notation used for analysis

Notation	Meaning
E_{tx}	Energy to send a data packet
E_{rx}	Energy to receive multiple data packets
P_{tx}	Power to transmit[a]
P_{rx}	Power to receive[a]
P_{idle}	Power in idle mode [a]
P_{sleep}	Power in sleep mode [a]
$P_{\text{wake-up}}$	Power to wake-up[a]
$\widehat{T}_i$	Cycle duration of a receiver r_i
T_r	RTR window period
T^i_{rw}	Maximum reception window of r_i
T^i_{idle}	Idle period at receiver r_i
T^{ij}_{idle}	Idle period at sender u_{ij}
T_{Data}	Time to transmit a data packet
T_{RTR}	Time to transmit a RTR packet
CW_{Data}	Contention window for data
CW_{RTR}	Contention window for RTR
slotTime	Time duration of a single slot
h	Number of hops
n	Number of wake-up in a cycle $\widehat{T}_i$
$D_{\max}$	Maximum end-to-end delay
$D_{\min}$	Minimum end-to-end delay
D_{avg}	Average end-to-end delay

[a]Power levels of CC2420 radio

5.1 Energy consumption

The performance of nW-MAC depends on the energy consumption for a number of RTRs, data and ACK sending and receiving, the number of wake-ups per cycle, the idle period in medium listening, etc. The expected energy to transmit a data packet by an upstream sender u_{ij} of receiver r_i, is calculated as

$$E_{tx} = (m \times T_{\text{RTR}}) \times P_{\text{rx}} + T_{\text{Data}} \times P_{\text{tx}} + T_{\text{ACK}} \times P_{\text{rx}} + T^{ij}_{\text{idle}} \times P_{\text{idle}}, \quad (6)$$

We assume that prior to sending a packet, u_{ij} has to listen for an expected duration of $T_r + \frac{\widehat{T}_i}{2n}$ and receive m number of RTRs from r_i. The later assumption is true when u_{ij} fails to transmit the data packet in response to a maximum $(m-1)$ number of RTRs; here, $(m-1)$ other transmissions are considered prior to u_{ij}'s data transmission. Thus, the idle listening time (T^{ij}_{idle}) at u_{ij} solely depends on m, such that

$$T^{ij}_{\text{idle}} = \frac{T_r + \frac{\widehat{T}_i}{2n}}{2} + (\text{SIFS} \times m) + \left(\text{SIFS} + \text{slotTime} \times \frac{\text{CW}_{\text{Data}}}{2} + T_{\text{Data}}\right) \times (m-1), \quad (7)$$

With a very low traffic, the value of m is expected to be small since the possibility of simultaneous senders at a given instant of time is minimal, and therefore there exists fewer contention and collision possibilities. In a single wake-up-based asynchronous staggered schedule, a sender listens to the medium for a maximum average bound of $\frac{\widehat{T}_i}{2n}$. When $n = 4$, this becomes approximately one eighth of the receiver's cycle $\widehat{T}_i$. It is worth mentioning that for a sender and a receiver in RI-MAC [1] and X-MAC [3], respectively, the average medium listening time is, analytically, one half of the cycle length; this is greater than that using nW-MAC.

Conversely, a receiver can receive multiple packets within the reception window. Thus, m data packets received at each Rx rendezvous require at least $(m+1)$ RTR transmissions, including the ACK for the last received packet. Hence, the energy for m packets received is

$$E_{rx} = ((m+1) \times T_{\text{RTR}}) \times P_{\text{tx}} + (m \times T_{\text{Data}}) \times P_{\text{rx}} + T^{i}_{\text{idle}} \times P_{\text{idle}}, \quad (8)$$

T^i_{idle} includes the average waiting time for a receiver prior to sending the first RTR. Additionally, an average of data contention back-off is considered after each of the m RTR packet transmissions. A maximum data back-off is also added to T^i_{idle} when r_i does not receive any data after sending the last RTR (with an additional data request).

$$T^i_{\text{idle}} = \left(\frac{T^i_{\text{rw}}}{2} \; or \; \frac{\frac{\widehat{T}_i}{2n}}{2}\right) + \left(\text{SIFS} + \text{slotTime} \times \frac{\text{CW}_{\text{Data}}}{2}\right) \times m + (\text{SIFS} + \text{slotTime} \times \text{CW}_{\text{Data}}), \quad (9)$$

The observations based on the energy analysis demonstrate that nW-MAC is able to maintain optimal energy consumption; in low traffic a receiver primarily wakes-up once per $\widehat{T}_i$ interval. Moreover, the energy cost is minimal since receivers avoid unnecessary medium listening [4], with the exception of the idle waiting time for data reception.

5.2 Delay optimality

To quantify the delay of nW-MAC, we analyze the minimum and maximum end-to-end delay distribution of a chain topology. In asynchronous staggered schedule, receiver r_i utilizes only one wake-up per cycle which is at the minimum delayed wake-up ($w_{k_{\min}}$). In such a schedule, the per hop delay is bounded by the maximum possible duration of the reception window T^i_{rw}. According to the start-of-cycle (SoC) selection in Section 4.2, the $w_{k_{\min}}$ wake-up of a receiver has, at most, a $\frac{\widehat{T}_i}{2n}$ long reception window. Additionally, the T^i_{rw} of r_i cannot exceed the downstream node d_i's wake-up offset ($t^{d_i}_{w_k}$). Therefore, the maximum end-to-end delay from a source node toward the sink is derived as

$$D_{\max} = \widehat{T}_i + (h-1) \times \left(\frac{\widehat{T}_i}{2n} - \frac{T^i_{rw}}{2}\right), \quad (10)$$

The assumption is made here that after generating a packet a source must wait, at most, for the duration of $\widehat{T}_i$ before sending it to the receiver. For the remainder of the $(h-1)$ hops the maximum delay should be $(\frac{\widehat{T}_i}{2n} - \frac{T^i_{rw}}{2})$.

Subsequently, if any source obtains an immediate rendezvous with the receiver after receiving the packet from the upper layer, the minimum delay would be

$$D_{\min} = h \times \left(\frac{\widehat{T}_i}{2n} - \frac{T^i_{rw}}{2}\right), \quad (11)$$

The delay approximation for adaptive multiple wake-up schedules is complex. Considering that to handle heavy traffic a receiver might need to wake-up a maximum of n possible times at each cycle (i.e., the worst-possible case) and accordingly announce its wake-up schedules or Rx rendezvous (using RTR). Being aware of the schedules, the upstream senders select all of the n wake-ups of the receiver as their Tx rendezvous. Eventually, all of the senders have an equal probability for sending data at each of the n wake-ups of the receiver. Hence, the per hop delay is equal to the cycle average which is $\frac{\widehat{T}_i}{2}$ and the average end-to-end delay is

$$D_{\text{avg}} = \frac{\widehat{T}_i}{2} + \frac{h-1}{2} \times (\widehat{T}_i - T^i_{\text{rw}}), \quad (12)$$

The delay analysis of nW-MAC demonstrates that in an asynchronous staggered schedule the per hop average delay is bounded by $\frac{\widehat{T}_i}{2n}$. Thus, for a chain the maximum per hop delay is approximately one eighth of the receiver's cycle time ($\widehat{T}_i$) ($n = 4$ for this paper's example). This causes a significant delay reduction as compared with the existing asynchronous RI-MAC [1] and X-MAC [3]; the per hop average delay for those protocols is one half of the cycle. Moreover, for a multiple wake-up schedule the average per hop delay of nW-MAC is equivalent to $\frac{\widehat{T}_i}{2}$, which contributes to minimizing the delay even under heavy traffic conditions.

6 Performance evaluations

6.1 Simulation setup

The simulations of nW-MAC are extensively performed in NS-2. The nW-MAC behavior for two different network conditions is studied: (1) single-flow performance in a chain topology, and (2) multiple flow performance in a tree topology. Routing traffic is avoided and a static path from each sensor node to the sink is used. The link bandwidth is set as 250 Kbps and all of the nodes in the network have a transmission and carrier sensing range of 25 and 55 m, respectively. Source nodes generate data either periodically or based on the detection of an event, or both.

Table 2 provides the details concerning the nW-MAC simulation parameters; the majority of them have been extracted from the NS-2 implementation of RI-MAC [1]. Furthermore, we use the power levels of CC2420 radio to measure the energy consumption. To gather knowledge on nW-MAC performance, asynchronous sender initiated X-MAC [3] and receiver

Table 2 Simulation Parameters

Parameter	Value
Channel bandwidth	250 Kbps
Transmission range	25 m
Carrier sensing range	55 m
Data packet size	32 bytes
RTR packet size	10 bytes
SIFS	192 μs
RTR window (T_r)	8 ms
Contention window ($\mathrm{CW}_{\mathrm{Data}}$)	16
Contention window ($\mathrm{CW}_{\mathrm{RTR}}$)	8
Slot time	320 μs
Retry limit	5
Simulation time	100 s

initiated RI-MAC [1] protocols are implemented in the same simulation environment and compared with *n*W-MAC. We use a 6 byte long strobe preamble for X-MAC which embeds the receiver's ID; X-MAC receiver remains active for 20 ms at each wake-up [3] and does a *clear channel assessment* (CCA) check for possible strobe preamble(s) from the sender. The CCA check continues for a time period longer than the interval between two consecutive preamble transmissions; a medium idle status at the expiration of the CCA check forces the receiver to sleep.

The cycle duration ($\widehat{T_i}$) of the receivers varies from 0.5 to 2.0 s. A single wake-up followed by a sleep period, in each cycle, is considered for the receivers in X-MAC and RI-MAC. However, for *n*W-MAC, provisions for multiple ($n = 4$) wake-ups are assumed for each receiver in every $\frac{\widehat{T_i}}{n}$ interval. In order to provide more stable results, an average of ten simulation executions is performed, each for 100 s, under the same network conditions.

For the simulation results, the following metrics are used to realize the performance of the proposed *n*W-MAC:

- *Energy Consumption*: the average energy consumption, in Joules, of all the nodes of the network.
- *Average Duty Cycle*: the ratio between per cycle average active time to the entire cycle time, expressed in percentage.
- *End-to-End Delay*: the time interval between a packet generated at the source and received at the sink.
- *Per Hop Delay*: it is calculated by dividing the end-to-end delay with the total number of hops.
- *Delivery Ratio*: the ratio between the total numbers of packets received by the sink to the number of packets generated by the source nodes.
- *Throughput*: it is calculated as the amount of received data (per second) by the sink over the simulation time.
- *Signaling Overhead (per packet)*: it is measured in bytes by dividing the total amount of bytes transmitted as control packets to the total number of successfully received packets at the sink node.

The error bars in the presented graphs show the variations of the obtained delay, and demonstrate the minimum and maximum values among the simulation runs.

6.2 Chain topology and single-flow performance

To attest to the concept of *n*W-MAC, it is very important to analyze the multi-hop performance in a chain topology. In this work's implementation, eight-node chain (seven hops) with one source and a sink at each end are used; the distance between two adjacent nodes is set to 20*m*.

Figure 7 shows the performance of the protocols for both energy and duty cycle. As we observe in Fig. 7a, the average energy consumption (measured at a rate of 2 pkts/s with a 1 second cycle) of X-MAC is higher than the other protocols, due to the preamble transmission causing more energy waste. The RI-MAC also consumes relatively more energy mainly for the huge idle listening at the sender end. However, the asynchronously scheduled rendezvous mechanism ensures less energy consumption for *n*W-MAC, as it reduces the idle listening at both the sender and receiver ends.

The graph of Fig. 7b presents the energy performance under different source reporting rates with a one second cycle. Although the energy consumption minimally increases at lower rates, as the rate increases the energy consumption becomes high for each of the protocols. At a high traffic rate, X-MAC has the highest energy consumption due to the frequent sending and receiving of strobe preambles. RI-MAC also has a high energy consumption due to the collision effect and unnecessary medium listening. At a high traffic rate, *n*W-MAC clearly outperforms the other protocols because of the scheduled wake-up and reception window strategy.

As shown in Fig. 7c, the duty cycle rises in proportion to the data rate. The X-MAC and RI-MAC result in a maximum of 52% and 34% average duty cycle, respectively. These are both much larger than the *n*W-MAC's 20% average duty cycle. The active time frequently increases for X-MAC, when multiple senders contend for medium access and only one of them is able to send the preamble or data. Moreover, in RI-MAC, receivers are mostly found awaken at high rates, though due

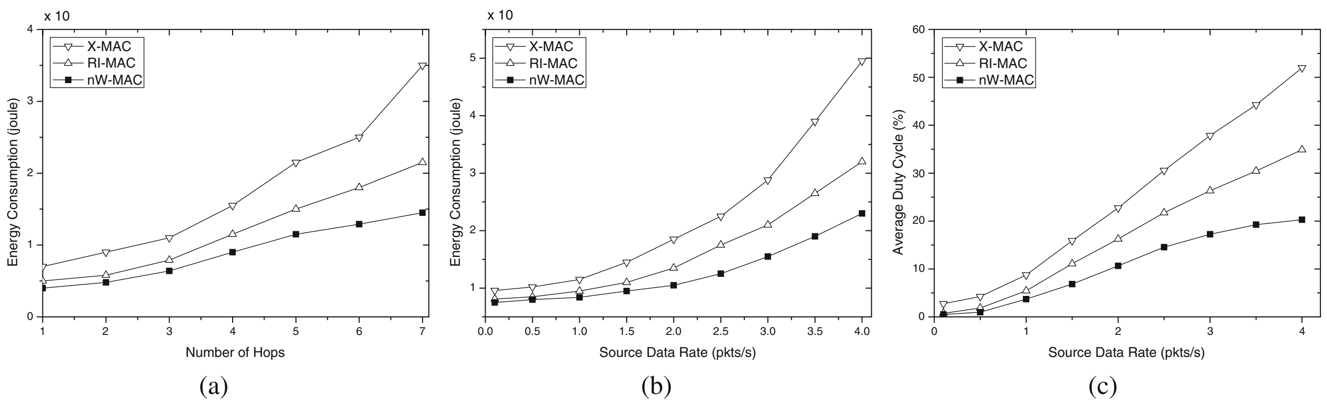


Fig. 7 Performance on energy and duty cycle for chain topology: **a** average energy consumption for different number of hops, **b** average energy consumption, and **c** average duty cycle for varying source data rate

to collisions their active time seems to be ineffective from the energy point of view. In contrast, *n*W-MAC allows a receiver to receive till the reception window that eventually bounds the active period of a node.

In Fig. 8, the end-to-end and per hop delay are evaluated. As shown in Fig. 8a, for all the protocols (with 2 pkts/s and one second cycle) the end-to-end delay increases as the number of hops in the chain increases. However, the delay of X-MAC and RI-MAC are severely affected by the sleep periods of the nodes. In *n*W-MAC, the delay is less than the other two protocols since all of the nodes are able to avoid sleep delay and adaptively use the provisions of multiple wake-up schedules for packet reception and sending.

Figure 8b demonstrates that in X-MAC, the overall delay increases due to the receiver's sleep schedule and medium occupancy time for the preamble transmission. Whereas, the sleep delay and increased retransmission due to uncoordinated multiple packet receipts mostly affect the RI-MAC's delay performance. Conversely, in *n*W-MAC, nodes receive multiple packets until the reception window limit and deliver data using the specified staggered or multiple wake-up schedule. The coordinated and bounded medium access mainly results in reduced end-to-end delay for *n*W-MAC.

The per hop average delay of the protocols are plotted in Fig. 8c; cycle durations 0.5 to 2 seconds are used with a low data rate of 0.5 pkts/s. The delay of *n*W-MAC is nearly identical to one-eighth ($\frac{1}{8}$) of the cycle, which is possible only for asynchronous staggered scheduling. In contrast, due to the randomness in the receiver's wake-up, the per hop average delay of RI-MAC and X-MAC are both approximately equal to one half of the cycle. The effectiveness of *n*W-MAC for a monitoring application is then verified with the delay performances demonstrated by the simulations.

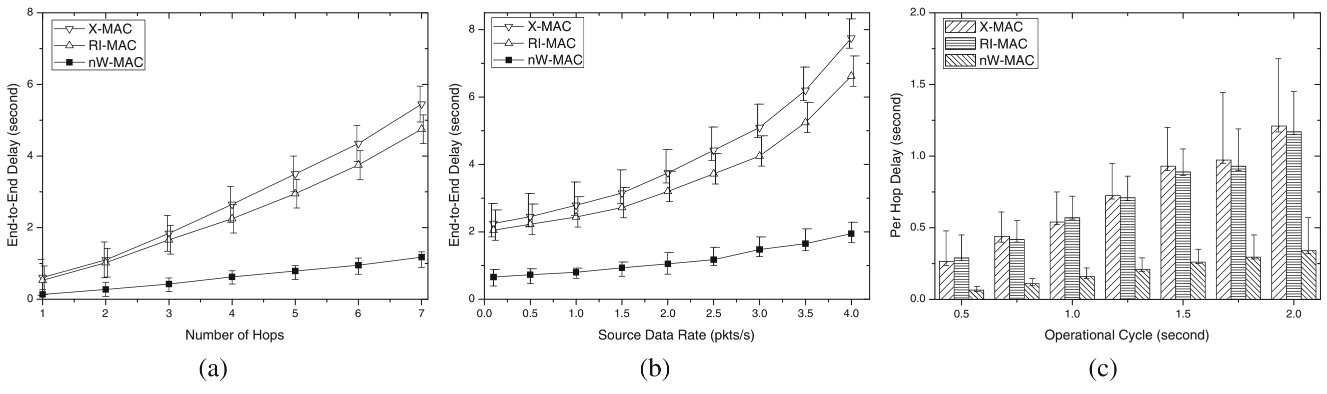


Fig. 8 Performance on delay for chain topology: **a** End-to-end delay for different number of hops, **b** end-to-end delay for varying source data rate, and **c** per hop average delay with respect to different operational cycle durations

6.3 Tree topology and multiple flow performance

A network of a 100 × 100 m area is used for the tree topology, where 100 nodes are deployed in a uniform random distribution. On average, most of the sensors are found within four to seven hops away from the sink.

In Fig. 9, the performances of basic *n*W-MAC and adaptive *n*W-MAC are compared with X-MAC and RI-MAC. To analyze the energy consumption, a low periodic traffic of 0.5 pkts/s is considered, along with a diverse number of sources. Figure 9a represents the general incremental trend of energy usage with an increased number of sources. When the number of sources is larger, X-MAC experiences significantly higher energy consumption than the other protocols. Note that for fewer sources, the basic *n*W-MAC consumes more energy, as compared with X-MAC, due to its initial scheduling having multiple wake-ups. However, as the number of sources increases, the amplification of the preamble transmission mostly affects X-MAC. Among the protocols, adaptive *n*W-MAC appears to be the most energy efficient since it ensures better energy utilization with minimum waste. RI-MAC consumes less energy for fewer sources; when the number of sources increases it consumes and wastes more energy due to a large number of collisions.

Adaptive *n*W-MAC achieves a lower duty cycle than X-MAC and RI-MAC, as shown in Fig. 9b. However, in case of fewer sources, basic *n*W-MAC has the highest duty cycle due to its fixed multiple wake-ups in each cycle. In X-MAC, prior to a data transmission a sender has to send the preamble while other senders in the same neighborhood wait in active mode; hence, the average duty cycle becomes longer. As the number of sources increases node participation in data forwarding also increases, resulting in a maximum 60% duty cycle for X-MAC. Conversely, in RI-MAC senders have to listen to the medium for a long period before sending the data; hence, the duty cycle is typically large. However, adaptive *n*W-MAC has the smallest duty cycle, 30% for 60 source nodes. The load-dependent scheduling and reception window-based MAC helps adaptive *n*W-MAC to achieve a superior duty cycle even with multiple wake-ups.

According to the results of Fig. 9c, the delay of *n*W-MAC surpasses each of the other protocols. Actually, the use of the multiple wake-ups as well as the multiple packet reception at each wake-up, results in steady delay behavior for *n*W-MAC, even with more sources. In contrast, the per cycle single wake-up strategy and sleep delay degrades the delay performances of both RI-MAC and X-MAC.

In Fig. 10, the tree topology performance of the protocols are compared for different source rates, where ten sources are randomly chosen. As shown in Fig. 10a, X-MAC responds with a greater energy consumption since the senders always attempt to be in rendezvous with their corresponding receivers. The basic *n*W-MAC also has a high energy consumption due to its unnecessary use of multiple wake-ups lacking traffic adaptation. Although RI-MAC maintains low energy consumption at low a rate, as the rate increases the energy consumption also increases for idle listening and collision related retransmission. In contrast, avoidance of such waste assures less energy consumption for *n*W-MAC even at a higher traffic rate. In simulations, it is observed that the number of contenders (or senders) decreases with an increase in the number of wake-ups in the same node cycle, resulting in fewer collisions in *n*W-MAC.

As demonstrated in Fig. 10b, during low rates, adaptive *n*W-MAC has a marginal duty cycle difference from both X-MAC and RI-MAC. However, as the rate

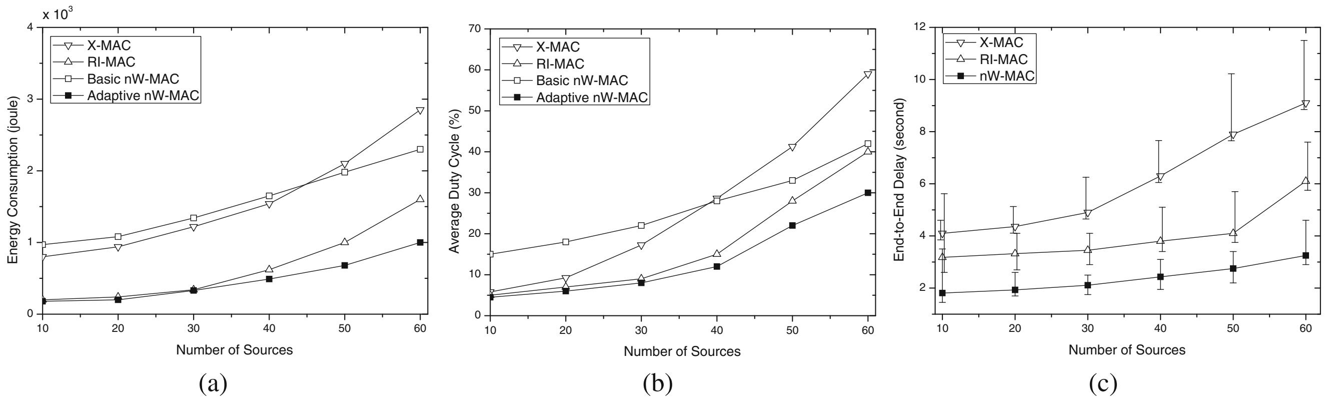


Fig. 9 Performance comparison for tree topology with different number of sources: **a** energy consumption, **b** average duty cycle, and **c** end-to-end delay

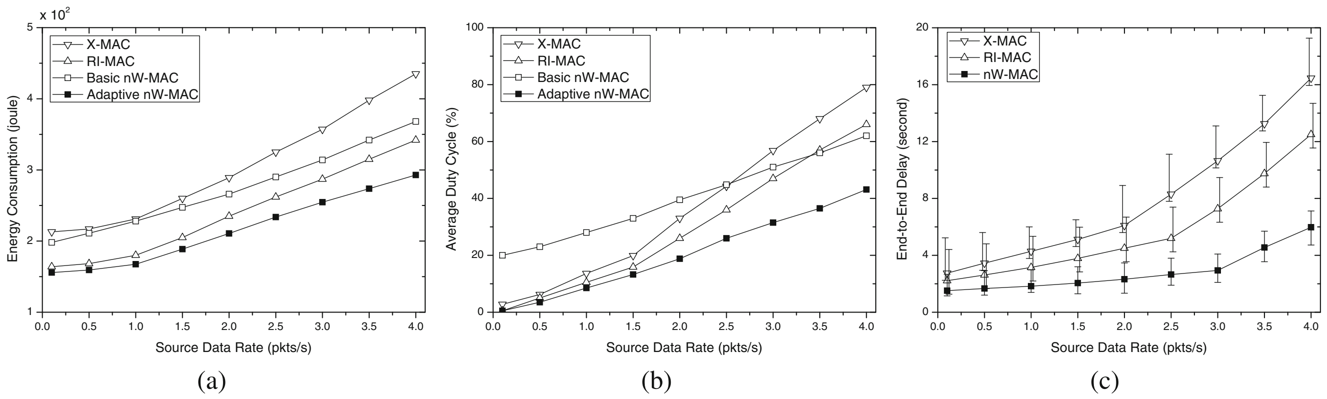


Fig. 10 Performance comparison for tree topology with different source traffic rates: **a** energy consumption, **b** average duty cycle, and **c** end-to-end delay

increases *n*W-MAC performs better due to its adaptability and efficiency in using *n* wake-up schedules. It is worthy to observe that at high traffic rates the duty cycle of basic *n*W-MAC is more concise than both X-MAC and RI-MAC, since the nodes in the basic protocol maintain a fixed wake-up(s) in each cycle.

The delay effectiveness of *n*W-MAC for different data generation rates is shown in Fig. 10c. As expected, the delay of *n*W-MAC is lower for the single wake-up strategy at a low traffic rate and the multiple wake-up strategy at a high traffic rate. In contrast, RI-MAC experiences a relatively high delay for typical sleep delay and collision related retransmissions at low and high traffic rates, respectively.

Figure 11 shows the delivery ratio of the protocols. Although at lower traffic rates the delivery ratio of RI-MAC and X-MAC demonstrate better performance, at high traffic rates their delivery ratio sharply decreases due to the increasing number of collision losses. For both low and high traffic, *n*W-MAC achieves a better delivery ratio; a maximum of approximately 95% of the generated packets is received by the sink during simulation. Thus, the results reflect the efficient use of multiple wake-ups in handling traffic variations.

Figure 12 compares the signaling overhead of the protocols as a function of the source rate, where it is observed that the signaling overhead of *n*W-MAC is less than the other protocols. In *n*W-MAC, the amount of contentions and collisions are always maintained at a sustainable level due to the adaptive and additive use of multiple wake-up provisions by a receiver at each cycle. Hence, it is possible to restrain the per packet signaling overhead (use of RTR and ACK) at an optimal level; which guarantees better throughput as well (shown in Section 6.4). In contrast, the lack of traffic adaptability incurs more collisions in RI-MAC,

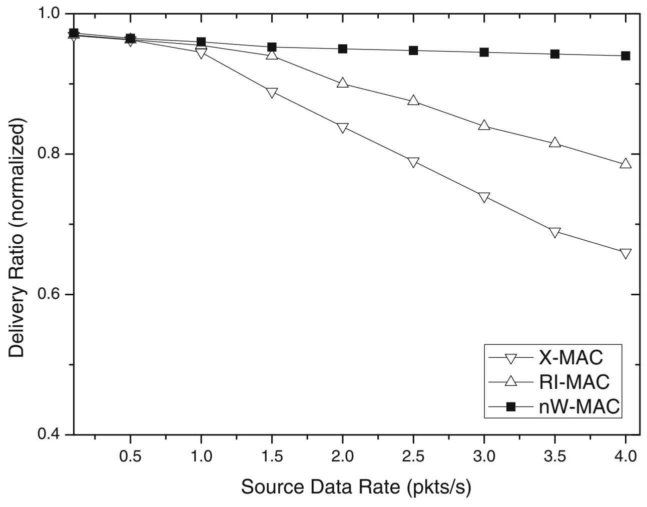


Fig. 11 Normalized delivery ratio for varying source rate

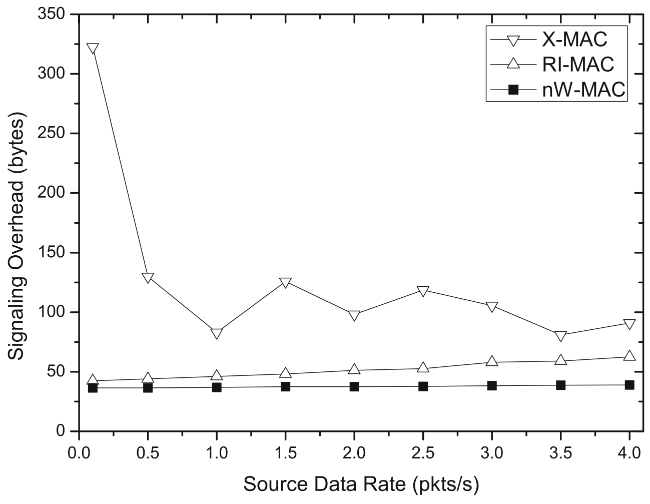


Fig. 12 Per packet signaling overhead

resulting in lesser throughput, especially at high traffic rates. Intuitively the contentions as well as per packet signaling overhead (use of beacon and ACK) are also increased for retransmissions. The overhead of X-MAC produces the poorest results, having more variations in the graph. This is because for a single data transmission the sender uses a variable series of strobe preambles until it receives an ACK from the receiver.

6.4 Multi-flow and multi-event performance

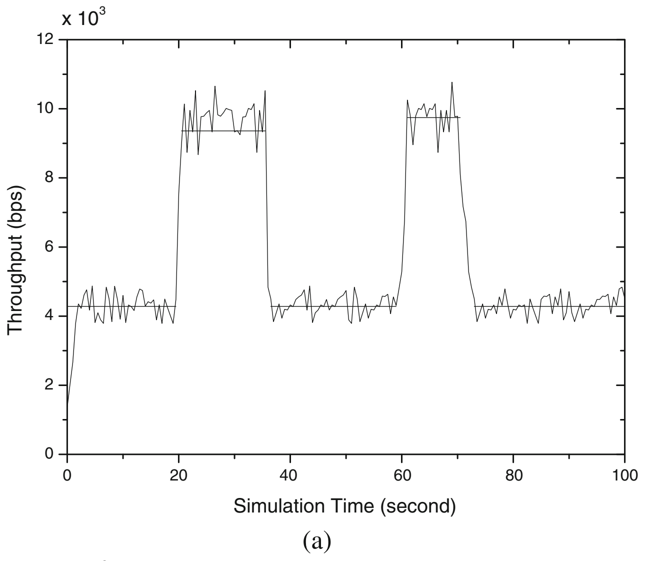
In order to investigate the applicability of *n*W-MAC for an event monitoring application, we further consider a simulation environment with regular traffic (periodic observation) at a very low rate of 0.2 pkts/s and event traffic (event detection) at a rate of 4 pkts/s. All of the nodes in the same tree topology (Section 6.2) periodically generate the regular traffic. In the network, ten sources in two randomly selected locations are considered for event data generation. The event traffic locations, triggering time, and maximum distance of the event sources from the sink are given in Table 3.

In Fig. 13, the throughput observations of both *n*W-MAC and RI-MAC are plotted over the simulation time, and the bar line shows the average throughput achieved at the sink. It can be seen that *n*W-MAC maintains a better throughput for both traffic environments. At low traffic, the use of asynchronous staggered schedule ensures an on time delivery of 88% of the packets every second. Whereas, for RI-MAC it is approximately 65% of the total generated packets. Furthermore, for event traffic, multiple wake-up schedules report 68% of the packets every second for *n*W-MAC, and RI-MAC ensures 45% reception of the packets.

We also observe the average end-to-end delay of the received packets during the event occurrence time. Figure 14 shows a sustainable average delay of 1.22 s for the packets of event 1, and 1.51 s for the packets of event 2. We believe that the achieved throughput and delay for *n*W-MAC are well suited for a monitoring network in order to fulfill the target of timely and effective actions.

Table 3 Event traffic settings

Event (id)	Location (x, y)	Time (s)	Distance (hop)
Event 1	(60, 80)	20th–35th	3
Event 2	(70, 15)	60th–70th	4



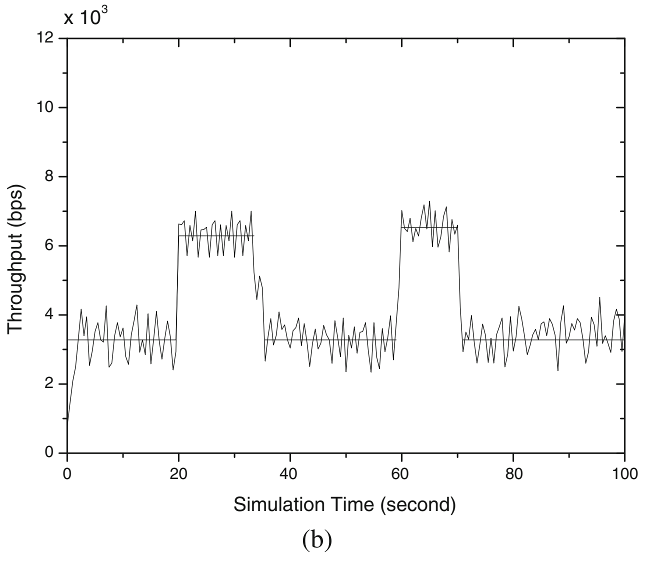


Fig. 13 Throughput performance: **a** *n*W-MAC and **b** RI-MAC

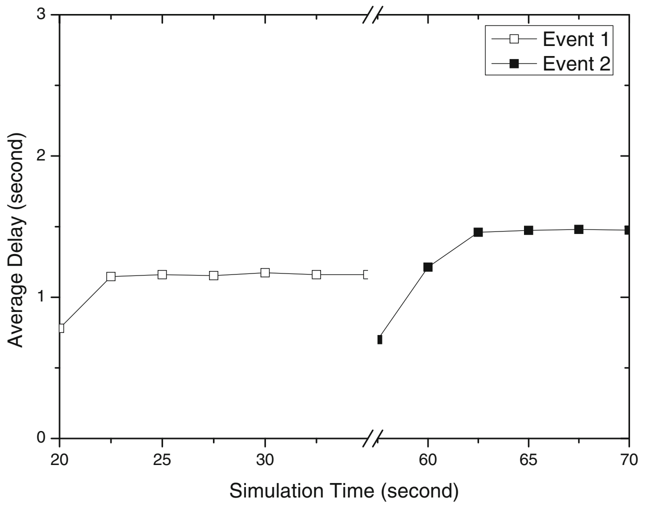


Fig. 14 Average delay of received event traffic

7 Conclusions

In this paper, we have proposed an innovative asynchronously scheduled duty cycle MAC protocol for WSN having n number of multiple wake-up provisions. Unlike previous MAC protocols, nW-MAC considers both energy consumption and delay, in parallel, and achieves a better energy-delay trade-off with multiple ($n = 4$) wake-ups per cycle for the receivers. Furthermore, reception window-based adaptive nW-MAC is able to cope with dynamic traffic environments, i.e., WSNs having monitoring applications. The analysis and simulation results demonstrate that nW-MAC has a lower energy consumption and better delay optimization, as well as throughput, than other currently existing protocols.

It is very challenging to incorporate broadcast traffic in any asynchronous duty cycle MAC, since each node maintains periodic wake-up and sleep schedules at random. Furthermore, the hidden terminal problem is a prospective research issue to be handled in sensor MAC protocols. Like existing asynchronous schemes, the nW-MAC protocol also carries these limitations, and we leave them for future research.

Acknowledgements “This research was supported by the MKE, Korea, under the ITRC support program supervised by the NIPA” (NIPA-2010-(C1090-1021-0003)). Dr. CS Hong is the corresponding author.

References

1. Sun Y, Gurewitz O, Johnson DB (2008) Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In: SenSys ’08: proceedings of the 6th ACM conference on Embedded network sensor systems. New York, NY, USA, ACM, pp 1–14
2. Polastre J, Hill J, Culler D (2004) Versatile low power media access for wireless sensor networks. In: SenSys ’04: proceedings of the 2nd international conference on Embedded networked sensor systems. Baltimore, MD, USA, ACM, pp 95–107
3. Buettner M, Yee GV, Anderson E, Han R (2006) X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In: SenSys ’06: proceedings of the 4th international conference on Embedded networked sensor systems. Boulder, Colorado, USA, ACM, pp 307–320
4. Ye W, Heidemann J, Estrin D (2004) Medium access control with coordinated adaptive sleeping for wireless sensor networks. IEEE/ACM Trans Netw 12(3):493–506
5. Lu G, Krishnamachari B, Raghavendra CS (2007) An adaptive energy-efficient and low-latency mac for tree-based data gathering in sensor networks: Research articles. Wirel Commun Mob Comput 7(7):863–875
6. van Dam T, Langendoen K (2003) An adaptive energy-efficient mac protocol for wireless sensor networks. In: SenSys ’03: proceedings of the 1st international conference on Embedded networked sensor systems. New York, NY, USA, ACM, pp 171–180
7. El-Hoiydi A, Decotignie J-D (2004) Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In: ISCC ’04: proceedings of the ninth international symposium on computers and communications 2004, vol 2 (ISCC”04). Washington, DC, USA, IEEE Computer Society, pp 244–251
8. Du S, Saha A, Johnson D (2007) Rmac: a routing-enhanced duty-cycle mac protocol for wireless sensor networks. In: INFOCOM 2007. 26th IEEE international conference on computer communications. Washington, DC, USA, IEEE Computer Society, pp 1478–1486
9. Sun Y, Du S, Gurewitz O, Johnson DB (2008) Dw-mac: a low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks. In: MobiHoc ’08: proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing. New York, NY, USA, ACM, pp 53–62
10. Ye W, Silva F, Heidemann J (2006) Ultra-low duty cycle mac with scheduled channel polling. In: SenSys ’06: Proceedings of the 4th international conference on Embedded networked sensor systems. New York, NY, USA, ACM, pp 321–334
11. Ahn G-S, Hong SG, Miluzzo E, Campbell AT, Cuomo F (2006) Funneling-mac: a localized, sink-oriented mac for boosting fidelity in sensor networks. In: SenSys ’06: proceedings of the 4th international conference on Embedded networked sensor systems. New York, NY, USA, ACM, pp 293–306
12. Tseng Y-C, Hsu C-S, Hsieh T-Y (2003) Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. Comput Netw 43(3):317–337
13. Thakur S, Nandi S, Bhattacharjee R, Goswami D (2007) An asynchronous wakeup power-saving protocol for multi-hop ad hoc networks. Int J High Perform Comput Appl 21(4):429–442
14. Wan C-Y, Eisenman SB, Campbell AT (2003) Coda: congestion detection and avoidance in sensor networks. In: SenSys ’03: proceedings of the 1st international conference on Embedded networked sensor systems. New York, NY, USA, ACM, pp 266–279
15. Rangwala S, Gummadi R, Govindan R, Psounis K (2006) Interference-aware fair rate control in wireless sensor networks. SIGCOMM Comput Commun Rev 36(4):63–74