

# YA-MAC: Handling Unified Unicast and Broadcast Traffic in Multi-hop Wireless Sensor Networks

Poonam Yadav, Julie A. McCann  
Department of Computing  
Imperial College London  
(pooyadav,jamm)@doc.ic.ac.uk

**Abstract**—This paper introduces YA-MAC, an agile Medium Access Control (MAC) protocol to provide high throughput for both unicast and broadcast traffic in Duty-Cycled Multi-hop Wireless Sensor Networks (DCM-WSN). YA-MAC is implemented under the UPMA framework in TinyOS and is evaluated on TelosB and MicaZ testbeds. We observe that YA-MAC significantly outperforms the state-of-the-art SCP-MAC protocol in terms of throughput by 60%, while tolerating a more dynamic network, at a small cost to duty-cycle performance. Further, we show that YA-MAC’s idle listening radio power consumption is 35% less than RI-MAC’s, while achieving similar throughput and latency.

## I. INTRODUCTION

Modern Wireless Sensor Networks (WSN) are required to handle both unicast and broadcast traffic. Unicast traffic, generally, consists of sensor data that is sent to the sink via an efficient route dictated by a routing protocol. Broadcast traffic consists of control messages such as time synchronisation messages or route update messages. The problem arises when we wish to maximise the lifetime for battery powered WSNs through duty-cycling. Here, sleeping nodes may miss broadcast messages thus impacting on the network’s ability to remain accurate and agile under changing conditions. Therefore, there is a requirement that the MAC layer efficiently provide a substrate for both unicast and broadcast traffic to support upper layer routing and dissemination protocols. However, this is challenging because the goals concerning each can conflict. Asynchronous communication has been shown to perform and scale well for unicast messaging because of the store & forward nature of many routing protocols [1]. Nevertheless, asynchronous MAC communication is not suited to broadcast traffic because message propagation requires co-ordinated wake-up schedules so that all nodes can receive the messages. Bar a few, most WSN MAC protocols are designed to handle either broadcast or unicast traffic for Duty-cycled Multi-hop WSNs (DCM-WSN). The MACs that support both do so at a significant cost to throughput compared with the more specialist schemes [2], predominantly due to additional overheads caused by time-synchronisation [3].

To address the throughput problem, we propose YA-MAC<sup>1</sup>, inspired by RI-MAC; a *receiver initiated* asynchronous MAC solely designed for unicast messaging that provides high throughput [4]. Aiming to efficiently handle simultaneous

broadcast and unicast traffic, YA-MAC is designed to avoid the overhead of time-synchronisation and explicit broadcast schedule exchanges. However, it uses an emergent broadcast slot mechanism (YA-EBS) in which each node coordinates its wake-up time slot<sup>2</sup> to allow broadcast message exchange within its single-hop neighbourhood. Therefore, it reaps the benefits of global synchronisation found in broadcast-friendly MACs, without having the disadvantages and complexity of requiring the whole network to be constantly globally synchronised after waking from sleep mode. Contributions of this paper are as follows:

- YA-MAC provides an integrated MAC protocol that efficiently handles unicast and broadcast traffic.
- YA-MAC improves sender node duty-cycling by predicting the intended receiver wake-up time, which in turn improves network-wide duty-cycling compared to the state-of-the-art *receiver initiated* asynchronous RI-MAC [4] duty-cycling.
- YA-MAC provides an emergent broadcast slot scheme with fast convergence and that maintains stability. It does not require explicit time synchronisation schemes therefore, provides improved duty-cycling and throughput.
- To the best of our knowledge, YA-MAC’s broadcast emergent slot scheme is the first approach of its kind that is implemented and evaluated on actual DCM-WSN testbeds.

## II. BROADCAST MESSAGE HANDLING SCHEMES IN DCM-WSN

To date there has been relatively little focus on broadcast message handling in duty-cycled networks. Such schemes can be broadly categorised as Flooding [5]–[7]; Gossip-like [8], [9]; and Single-hop Local Discovery (SLD). Flooding and Gossip, also known as opportunistic broadcast, schemes are typically used for network-wide dissemination of commands, configurations, code binaries, etc. SLD schemes typically propagate local (1-hop) information; disseminating link qualities, hop-counts and time information required by routing algorithms or time-synchronisation algorithms respectively. However, given that SLD broadcast schemes can also implement Flooding and Gossip mechanisms (where messages

<sup>2</sup>In YA-EBS, each node has only one broadcast slot in a single broadcast interval

<sup>1</sup>Yet Another MAC

are rippled out across the network) we concentrate on SLD broadcast specifically. Synchronisation of wake-up slots is crucial to SLD operation for DCM-WSN, and can take the following forms:

#### A. Network-wide Synchronised Broadcast Slots(NSBS)

A NSBS approach is straightforward but requires global synchronisation, e.g., S-MACL [10]. That is, every node conforms to a single common NSBS, each is aware of the shared slot in which each node will wake up and exchange broadcast messages. The size of slot therefore, should be large enough to accommodate all message exchanges across the network. Within this, nodes either asynchronously broadcast messages (CSMA) or opt for TDMA based scheduling schemes to avoid contention and hidden terminal problems [3].

#### B. Local one-hop Synchronised Broadcast Slot(LSBS)

The LSBS scheme is based on the formation of virtual clusters where all nodes in a one-hop virtual cluster have a common time period for broadcast, e.g. S-MAC [11] uses this scheme to accommodate both broadcast and unicast traffic. This scheme works well in managing synchronisation for a single-hop neighbourhood but falls down in topologies where nodes form part of more than one cluster. Such nodes are required to wake up at different time slots to receive messages from each cluster in turn. This too has congestion implications.

#### C. Broadcast messages as Unicast messages

In Asynchronous MACs such B-MAC [1], X-MAC [12], and RI-MAC [4], broadcast messages are handled in the same way as unicast messages. This scheme is expensive from the sender's point of view, as it must remain awake for the complete duty-cycle period to ensure that all receivers receive the message. Further, the sender has to send the given message a number of times because receivers may have different wake-up schedules, hence increasing the number of transmissions.

#### D. Our Approach- YA-EBS: An Emergent Broadcast Slot

The above approaches do not efficiently handle SLD broadcasts. To address the high contention (found in NSBS, refer-II-A); the high duty-cycles with multi-slot scheduling (found in LSBS, refer-II-B); and the multi-transmissions per single broadcast message (found in II-C), we present YA-EBS - an Emergent Broadcast Slot Scheme<sup>3</sup>. In YA-EBS a Synchronisation Error Tolerance Window (SETW) (refer to definition-1) for a node, emerges such that it partially overlaps the SETWs of its neighbourhood nodes(see Fig-1). This loose synchronicity reduces overheads and contention in the network. A node then transmits its broadcast message half-way through this window, which is received by the nodes currently in their SETWs. Also, as the convergence and stability of such emergent schemes are affected by unreliable wireless links, instead of requiring that all nodes have established their SETW before proceeding, only a sub-set are used; introducing the

<sup>3</sup>Though inspired by the Meshed Emergent Slot Synchronisation (MEMFIS) [13] and RFA, YA-EBS exhibits better performance [14].

notion of a 'degree' of *Synchronicity* (refer to definition-2). Further convergence optimisations in YA-EBS involve a fast phase advancement technique thus:

$$\phi'(t^+) = \begin{cases} \phi'(t) & \text{if } ((1 - \varepsilon) \leq \phi'(t)) \parallel (\phi'(t) \leq \varepsilon) \\ g(\phi'(t)) & \text{if } (\varepsilon \leq \phi'(t) \leq (1 - \varepsilon)) \end{cases} \quad (1)$$

Here,  $\phi'(t^+)$  is new phase value,  $\phi'(t)$  is previous phase value,  $g(\phi'(t))$  is phase advancement function, i.e.,  $g(\phi'(t)) = \frac{\varepsilon}{1-\varepsilon}\phi'(t)$ ; such that  $\varepsilon \in (0, 0.5)$ . Here,  $2 \times \varepsilon$  is node's wake-up time fraction in duty-cycle time period,  $T$ .

**Definition 1.** *The Synchronisation Error Tolerance Window (SETW) is the period of time within the regular broadcast time interval,  $T$ , such that  $SETW = [-\varepsilon T, \varepsilon T]$ ,  $\varepsilon \in (0, 0.5)$  and its width,  $W = 2 \times \varepsilon \times T$ .*

A node which is part of more than one virtual cluster, exchanges broadcast messages with all its one-hop neighbours during the SETW. Recall, a node in a LSBS scheme must maintain different time slots for each cluster, only a single slot (SETW) is required by YA-EBS. Further, though the width of SETW may be larger than the LSBS slots, YA-EBS maintains low duty-cycles for those particular nodes, as observed in Motelab experiments (See figure-4).

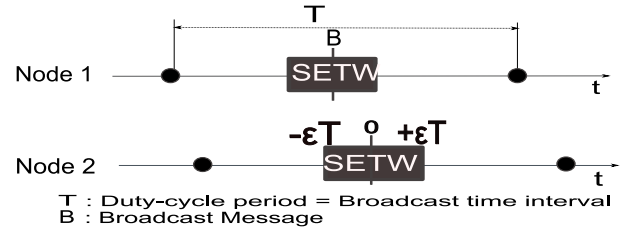


Fig. 1. YA-EBS Broadcast handling mechanism. Once YA-EBS converges, nodes awake only in their respective SETWs.

**Definition 2.** *The Synchronicity of a node in a DCM-WSN is the percentage of the total number of neighbours whose broadcast messages it can hear during its SETW.*

### III. PROBLEM ANALYSIS

To handle broadcast traffic, one is required to optimise the throughput and duty-cycling (maximizing sleep). In this section, we formulate this problem to better understand which of configuration parameters will have the biggest impact on broadcast message performance. In doing so, we also show that YA-MAC is the best approach.

#### A. Network Models and Assumptions

We model the DCM-WSN as a graph,  $G(S, L)$ , that is composed of  $S$  sensor nodes represented by set  $S = (1, 2, \dots, S)$  where  $L$  is a set of all transmission links between these  $S$  sensor nodes, i.e.;

$$L = \bigcup_{s_i, s_j \in S} \langle s_i, s_j \rangle \quad (2)$$

Here,  $\langle s_i, s_j \rangle$  define a link iff node  $s_i$  is within the transmission range of node  $s_j$ .

The m-hop neighbourhood  $N^m$  is composed of the node itself and its m-hop neighbours. Thus, the 1-hop neighbourhood of the node  $d_i$  is given by:

$$N_i^1 = \{s_i\} \cup \left( \bigcup_{\langle i,j \rangle \in L} \{s_j\} \right) \quad (3)$$

The local degree of connectivity of node  $s_i$  also can be represented by  $deg_i = \|N_i^1\|$ . The average degree of connectivity for the whole network can be represented as:

$$\bar{deg} = \frac{\sum_{i=1}^S \|N_i^1\|}{N} \quad (4)$$

We assume that every node sends a broadcast message asynchronously after a periodic time interval,  $T$ .  $\gamma_i$  represents the number of times a node  $n_i$  wakes up<sup>4</sup> to receive scheduled broadcast messages from its one-hop neighbourhood nodes  $n_j \in N_i^1$  during  $T$ . Assuming that the node wakes up  $\gamma_i$  times in  $T$ ,  $\widehat{\omega}_k^i$  represents the fraction of the time it is awake during  $k^{th}$  wake-up in  $T$ . Therefore, the total wake up time in  $T$  for receiving or sending all the broadcast messages can be represented as:

$$\omega^i \times T = \sum_{k=0}^{\gamma_i} \widehat{\omega}_k^i \times T \quad (5)$$

Also,

$$\omega_{Idealmin}^i = deg_i \times \tau \quad (6)$$

Here  $\tau$  is the minimum time fraction of  $T$  required for node  $n_i$  to receive a broadcast message from its neighbourhood.  $\omega_{Idealmin}^i$  represents the minimum time a node  $n_i$  needs to receive broadcast packets sent during time  $T$  by its one-hop neighbourhood, assuming all neighbours are perfectly time scheduled and there are no collisions.

Next, we define the transition time fraction  $\eta$  as the time a node takes to switch ON and OFF, with respect to  $T$ . Therefore, for  $\gamma_i$  wake-ups, the transition time is  $\gamma_i \times \eta \times T$ . We also define the threshold for  $\gamma_i$  as  $\gamma_i^{Th}$ , with the assumption that a node wakes up to hear broadcast messages from each of its one-hop neighbours in  $T$ ,  $\gamma_i^{Th} = deg_i$ .

**Goal 1:** Now we begin with the formulation of our first goal, i.e., the minimisation of the wake-up time duration of a node  $n_i$  in  $T$ ; mathematically represented by:

$$\Psi(\gamma_i, \omega^i) = (\gamma_i \times \eta) + \omega^i \quad (7)$$

Further, this can be described as an optimisation problem such that:

$$\begin{aligned} &\text{Given: } T, deg_i \\ \mathbf{P1} : & \text{Min}_{\gamma_i, \omega^i} [\Psi(\gamma_i, \omega^i)] \end{aligned} \quad (8)$$

<sup>4</sup>That is, the transition from OFF to ON, which has associated warm up costs

$$\begin{aligned} \text{S.T} \quad & 1 \leq \gamma_i \leq \gamma_i^{Th} \\ & \Psi(\gamma_i, \omega^i) \leq 1 \\ & \omega_{Idealmin}^i \leq \omega^i \leq 1 \end{aligned}$$

**P1** represents the LSBS and broadcast as unicast schemes due to the fact that node wakes up few times (say  $\gamma_i$  that is greater than 1) to listen for broadcast messages from its 1-hop neighbourhood nodes. Further, in NBBS and YA-EBS the node wakes up only once in  $T$  to listen to broadcast message from its 1-hop neighbourhood nodes. Therefore, for latter schemes, we set  $\gamma_i = 1$  that decompose **P1** into **P2** as given below:

$$\mathbf{P2} : \text{Min} [\omega^i] \quad (9)$$

$$\text{S.T} \quad \omega_{Idealmin}^i \leq \omega^i \leq 1$$

**Goal 2:** Now we formulate our second goal, i.e., the minimisation of the number of broadcast transmission<sup>5</sup> for  $n_i$  in  $T$ , and subsequently for the whole network. We aim at improving broadcast throughput by minimizing the redundant broadcast transmissions that lead to wireless channel occupancy. The function  $\Upsilon(z)$  represents the minimum average number of broadcast transmissions a sender node has to perform in a SLD scheme,  $z$ , to deliver a single broadcast message to all its 1-hop neighbours. Therefore, our aim is to find the best scheme (among those discussed in section II) that minimises the broadcast transmissions for a given environment, mathematically represented by:

Given:  $T, deg_i$

$$\mathbf{P3} : \forall z \text{ Min} [\Upsilon(z)] \quad (10)$$

$$\text{S.T} \quad z \in \{\text{SLD Broadcast Schemes}\}$$

Inferring from sections II-A to II-D, we state that **P3** will achieve an optimal value when  $z$  is either NSBS or YA-EBS. That is, a sender in both LSBS and broadcast-as-unicast schemes will be required to send a broadcast message more than once. Furthermore, if we consider factors such as network-wide time synchronisation, and contention due to hidden terminal problems for NSBS, the node  $n_i$  must requires a large time slot  $\omega^i$  that proportional to the 2-hop neighbourhood size for all nodes instead of only 1-hop neighbourhood size (to overcome contention due to hidden terminal problems). Therefore, we can say that  $\omega^i$  for all NSBS schemes must be greater than that of YA-EBS. However, optimal value of  $\omega^i$ , known as the SETW, requires further exploration, see section-V-A1 later.

#### IV. OVERVIEW OF YA-MAC OPERATION

The aim of YA-MAC is to incorporate both unicast and broadcast traffic simultaneously in the network while maximizing sleep times and throughput. In YA-MAC each node will be in one of the three phases: the *initialisation phase*, the *broadcast slot emergence phase*, and the *steady duty-cycled phase*, at any instance of time.

<sup>5</sup>In other words, the minimisation of packet loss due to collisions, contention, etc.

### A. Initialisation Phase

In the *initialisation phase*, all nodes in the network are 100% duty cycled. This sounds expensive, however, for the most part this will occur once (or very infrequently). Here nodes start their random timer and initialise parameters such as the broadcast time interval ( $T$ ) and the periodic unicast message *request-for-send* interval ( $t_{req}$ ). The values of these QoS parameters are communicated from the upper layers of the DCM-WSN stack. The broadcast timer with period  $T$  is thus started as soon as the random timer fires. In this time period,  $T$ , nodes calculate their one-hop neighbourhood size by listening the broadcast message while keeping themselves awake in this time period as shown in Fig-2. Note that, YA-MAC is not maintaining its own neighbourhood table but only uses the neighbourhood size that is calculated on the basis of number of broadcast messages received in single broadcast message interval,  $T$ , in the initialisation phase. However, the routing layer's link estimation mechanisms can be used to make precise and accurate neighbourhood size estimation.

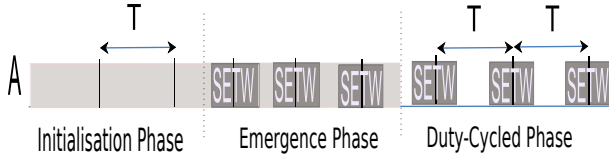


Fig. 2. A node is 100% Duty-cycled in the Initialisation phase and Emergence phase and wake-up only in its SETW during the Duty-cycled phase.

### B. Broadcast Slot Emergence Phase

Recall that the *Broadcast slot time emergence phase* is entered as soon as nodes calculate their neighbourhood size by keeping themselves 100% duty-cycled. Here each node tries to coordinate its SETW with its neighbours' using the YA-EBS scheme as discussed in section-II-D( for details refer to [14]).

**Definition 3.** A node's *Synchronicity Threshold* is defined as the minimum percentage of nodes from its neighbourhood that it can hear during its SETW.

### C. Steady Duty-Cycle Phase

The steady duty-cycle phase occurs when a given node's *Synchronicity* (refer to definition-2) becomes more or equal to the *Synchronicity threshold* (refer-definition-3)<sup>6</sup>. The value of the *Synchronicity threshold* is set by the upper layers in the WSN stack and represents a degree to which the network can tolerate missing broadcast messages. Note that higher value of the *Synchronicity threshold* represents a higher broadcast message throughput at the cost of duty-cycle performance. Its affect on the network in terms of message delivery ratios and duty-cycles is evaluated in section-V. In this paper we consider a uniform *Synchronicity threshold* for all nodes in the network. However, YA-MAC does not preclude more gradient-based or dynamic schemes.

<sup>6</sup>Note: *Synchronicity threshold* is represented as Synchronicity(%) or Local Synchronicity in the subsequent experiments.

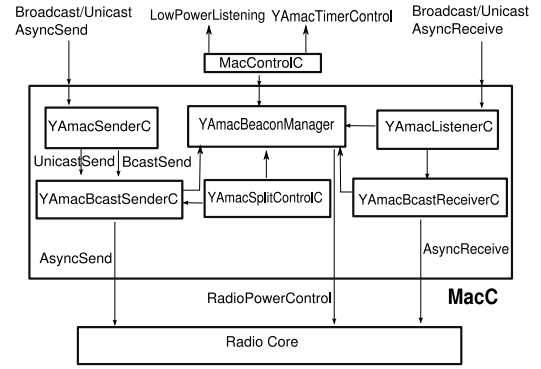


Fig. 3. YA-MAC TinyOS Implementation

The nodes are now in duty-cycle mode with a sleep interval dictated by the upper layers. For unicast messages, YA-MAC works in a *receiver-initiated* manner by first broadcasting a *message receive request* or *req-beacon* (as per RI-MAC [4]). Sender nodes in RI-MAC must remain awake to avoid missing the receiver. To avoid this high duty-cycle requirement, YA-MAC's sender node predicts the receiver wake-up schedule, based on the schedule history, and adjusts its duty-cycle accordingly to listen to the *req-beacon* from the receiver. YA-MAC also provides a collision avoidance mechanism. To achieve this, the receiver node embeds the number of nodes in its 1-hop neighbourhood into the *req-beacon*. The sender nodes use this and the length of time that the node has waited to send, to calculate its priority to avoid collisions. For broadcast messages, all nodes periodically wake up for the SETW and exchange their SLD broadcast messages. A node whose *Synchronicity* value falls below the *Synchronicity threshold* switches back to the *Broadcast slot emergence phase*; returning to being 100% duty-cycled. The various reasons for a drop in a node's *Synchronicity* value can be due to clock drift, a change in network density, etc. This method of switching back has the advantage that, unlike gradient based schemes [15], one node's disturbance does not perturb the whole network; therefore, our scheme is both agile and tolerant to change.

### D. YA-MAC Implementation

We implemented YA-MAC under the UPMA [16] framework in TinyOS 2.x for the CC2420 radio, compliant to *IEEE 802.15.4* standards with a data rate of 250 kbps. Fig-3 shows the component architecture of YA-MAC within the generic MacC configuration provided by the UPMA framework. YA-MAC's MacC component provides the YAmacTimerControl interface to the upper layers that can directly control the broadcast time interval and *req-beacon* time interval. YAmacBcastSenderC has responsibility of sending broadcast messages, unicast messages, *req-beacon*, and duty-cycle timers. YAmacBcastReceiverC has the responsibility of extracting information from the incoming *req-beacon* regarding congestion back-off and event-handling data. That is, when a receiver node widens its adaptive listening window, it declares this to

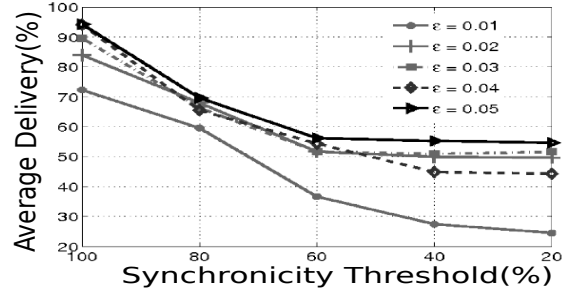
the senders by setting a binary flag in the *req-beacon*. Further, the YA-MAC emergent broadcast scheme implementation interacts between both these components via shared timers through *Timer Interfaces* and the *SendReceive Interface*, which allows the *YAmacBcastReceiverC* component to initiate data sent after it receives the *req-beacon* from the receiver. Unlike implementations of RI-MAC, we have not made any changes to the UPMA Radio core component; therefore, YA-MAC is compliant with other UPMA components and can be applied more generally.

## V. EVALUATION

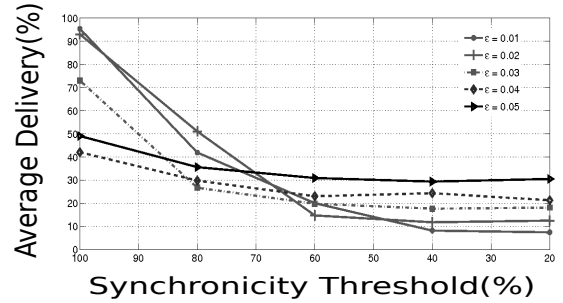
We perform our evaluation experiments on three different size testbeds: Motelab (87 Telosb nodes) [17], NUS (124 Telosb nodes) [18], and our local lab (10 MicaZ nodes). We conducted experiments in two sets. The first set of experiments evaluates the convergence of YA-MAC's SETW. Our ultimate aim is the maximisation of the broadcast message delivery ratio while minimizing duty-cycles (as discussed in III). To achieve this, for a given environment, and broadcast time interval ( $T$ ) dictated by the application, we wish to find the appropriate  $\epsilon$  and *Synchronicity threshold*. We also use these values to gage YA-MACs performance in terms of comparative schemes. We begin our experiments by fixing the value of the broadcast time interval,  $T$  and varying the  $\epsilon$  and the *Synchronicity threshold*, and measure duty-cycle and throughput performance. We then examine the behaviours resulting from varying the network topology and density. The second set of experiments aim at evaluating YA-MAC for both unicast and broadcast traffic. In the experiments we use fix values of  $\epsilon$  and the *Synchronicity threshold*, as derived from first set of experiments, and then vary the unicast and broadcast traffic rates respectively.

### A. Convergence Evaluation

1) *YA-EBS Convergence Evaluations*: These experiments are designed to evaluate the affect the *Synchronicity threshold* and SETW have on the average message delivery ratio and average duty-cycle for a given network. The experiments are executed on Motelab with a broadcast interval time ( $T$ ) set at 10 seconds (representing typical routing control messages). We initially ran the experiment with all nodes at 100% duty-cycle to ascertain the upper bounds of message delivery achievable in that environment, to allow us to calculate delivery ratios. Recall the SETW width  $W = 2 \times \epsilon \times T$ , for this experiment  $\epsilon$  varies from 0.01 representing 2% duty-cycle, to 0.05 which is 10% duty-cycle. For each run, we measured the average number of broadcast messages delivered, and then we vary the *Synchronicity threshold* from 100% to 20% in 5 steps. When the *Synchronicity threshold* is 100%, we observe that for all  $\epsilon$  values, there is a high average duty-cycle performance, Fig-4. This is because the nodes are not switching to the steady duty-cycled phase, hence remaining in the slot emergence phase for much of the time. At a 60% synchronicity threshold for all  $\epsilon$  values, we observe a 45% drop in throughput but an improvement of 20-25% in duty-cycle performance. Therefore,



(a) Average Delivery

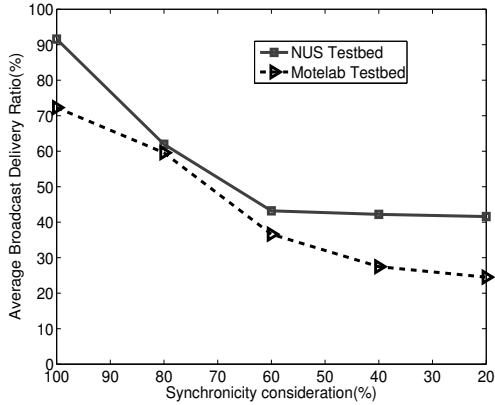


(b) Average Duty Cycle

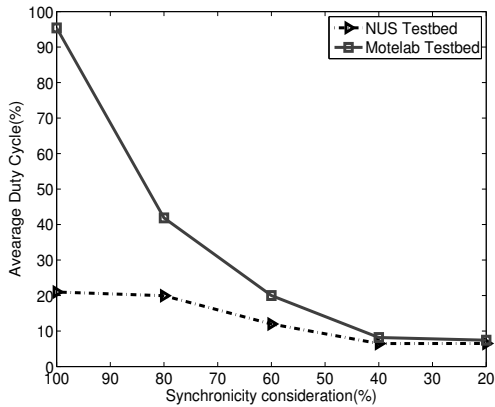
Fig. 4. YA-MAC synchronisation algorithm performance on 87 telosb random and dense testbed.

given the strong trade-off between throughput and duty-cycle the choice of settings must reflect whether battery lifetime or message delivery is important. Lower values of  $\epsilon$  equate to a smaller SETW in which the nodes have to exchange their broadcast messages. If  $\epsilon$  is large there is more chance that nodes converge faster. However, after the initialisation phase this window is unnecessarily large thus negatively impacting energy consumption and latency. Similarly, we observe a higher duty-cycle for  $\epsilon = 0.05$  than 0.01 for when the *Synchronicity threshold* is 20-40% .

2) *YA-EBS with varied Network Topology and Density*: Fixing the *Synchronicity threshold* to 80 and 60% and  $\epsilon$  to 0.01 and 0.05, we examine the effect of testbed (topology) and specifically node density. These experiments are evaluated on both the Motelab and NUS-Indriya testbeds. Interestingly, both testbeds have an average degree of connectivity of 13-14. We set  $\epsilon = 0.01$  with  $T = 10$  seconds. Compared with Motelab, we found that the NUS testbed delivers a 20% better message delivery and lower duty-cycles of about 20%, for both *Synchronicity thresholds* as shown in Fig-5. Further, YA-EBS converges very quickly even with a high *Synchronicity threshold* on this. However, the same experiments on Motelab, with a similar average node density, exhibit higher duty-cycles at 95% and 45%, for *Synchronicity thresholds* of 100% and 80% respectively. We believe this is because of the some nodes in Motelab, those who have shown extremely high connectivity



(a) Average Delivery Ratio



(b) Average Duty Cycle

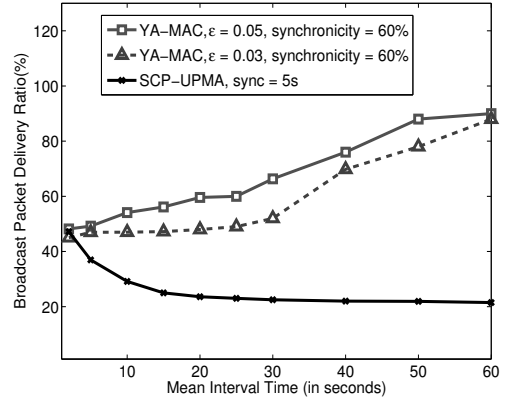
Fig. 5. Performance comparison of YA-MAC broadcast message delivery on two testbeds (Motelab and NUS) with average degree of connectivity 13-14 and with broadcast time interval,  $T = 10$  seconds, and  $\epsilon = 0.01$

(nearly 25 or more) took more time to converge.

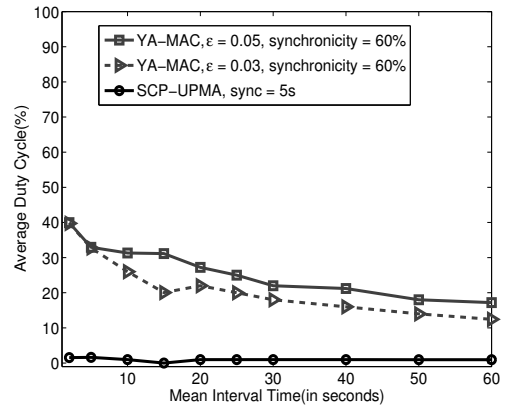
### B. Comparing YA-MAC to RI-MAC and SCP-MAC

We have now observed the effects of  $\epsilon$  and the *Synchronicity threshold*, for our next set of experiments we focus on the Motelab testbed only, and fix  $\epsilon = 0.02$  to  $0.05$  and  $80$ - $40\%$  *Synchronicity threshold*.

We compare YA-MAC with the state-of-the-art, SCP-MAC [2] and RI-MAC [4] schemes. SCP-MAC has designed specifically to optimise both unicast and broadcast messaging while maintaining good duty-cycle performance, exhibits low throughput performance; around  $20\%$ . RI-MAC, specifically designed for unicast traffic only, shows high throughput; up to  $100\%$ . Therefore, the aim of YA-MAC is to achieve comparable high throughput performance to RI-MAC while supporting both unicast and broadcast messages. Acknowledging, that this high throughput may cost duty-cycle performance, we aim that YA-MAC's duty-cycling should not be worse than  $10\%$  of SCP-MAC's on average.



(a) Average Delivery Ratio



(b) Average Duty Cycle

Fig. 6. Performance Comparison of YA-MAC and SCP-UPMA on a random network of 86 Telosb nodes, for broadcast traffic where each node sends a broadcast time interval ( $T$ ) varying from 1 to 60 seconds. For broadcast messages,  $\epsilon = 0.05$  and the *Synchronicity threshold* =  $60\%$ .

1) *Broadcast Traffic Analysis*: Examining broadcast messages with an  $\epsilon = 0.05$  and  $\epsilon = 0.03$  and a *Synchronicity threshold* of  $60\%$  we compare YA-MAC with SCP-MAC and RI-MAC. Initially we wish to vary the value of the broadcast time interval,  $T$ , from 5 to 60 seconds to reflect typical routing algorithm values. Fig-6(a) and Fig-6(b) shows that for YA-MAC, varying  $\epsilon$  shows little difference in throughput and duty-cycle performance. The effect of a larger  $T$  is intuitive, producing lower duty-cycles and higher throughputs. Comparing YA-MAC with SCP-MAC we observe YA-MAC is significantly better for larger values of  $T$ , and when compared with SCP, YA-MAC's throughput is superior by  $25\%$ - $70\%$ , see Fig-6(a) and Fig-6(b), however this comes with a small cost in duty-cycle percentages. Yet, the duty-cycle percentage improves with increased  $T$  for YA-MAC; therefore, where battery lifetime is of utmost importance, one would choose a lower  $\epsilon$ .

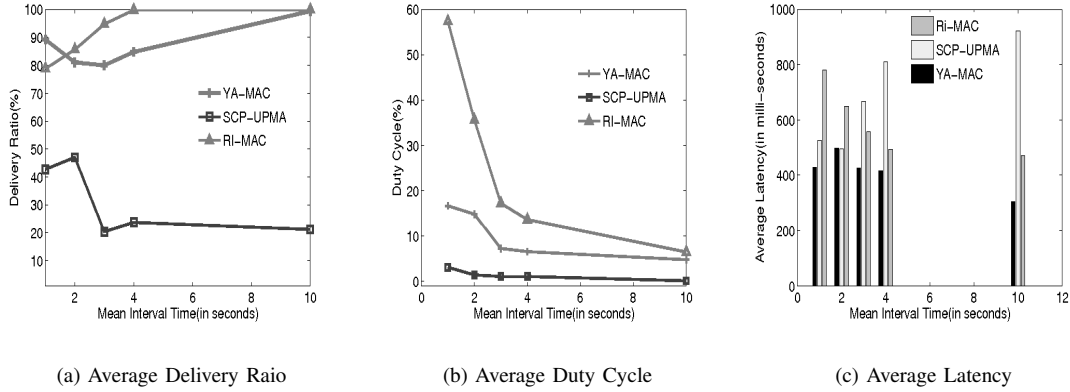


Fig. 7. Performance Comparison of YA-MAC, RI-MAC and SCP-UPMA on a random network of 47 Telosb nodes, for unicast traffic where each node sends and receives from 1 packet per 2 seconds to 1 packet per 10 seconds.

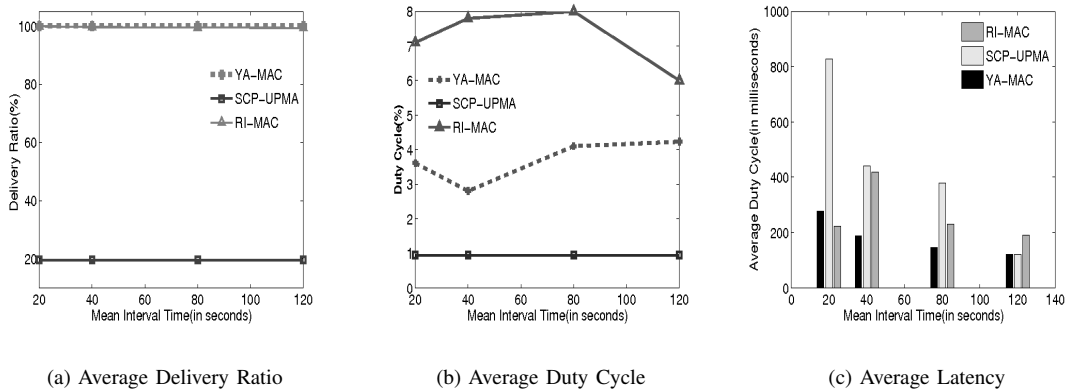


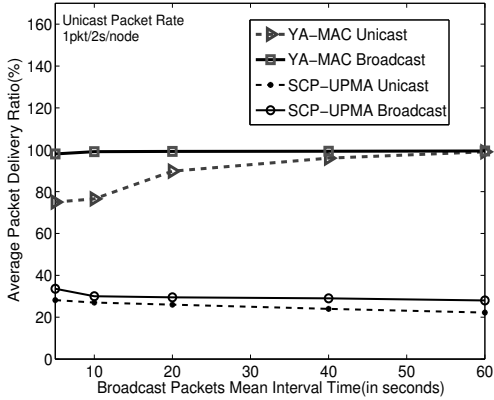
Fig. 8. Performance Comparison of YA-MAC, RI-MAC and SCP-UPMA on a random network of 47 Telosb nodes, for unicast traffic where each node sends and receives 1 packet every 20 seconds to 120 seconds. SCP-UPMA channel polling interval = 1 second and RI-MAC and YA-MAC *receiver-req* beacon interval = 1 second.

2) *Unicast Traffic Analysis*: We examine unicast messages using a receiver *req-beacon* interval of 1 second for YA-MAC and RI-MAC. To normalise the settings we kept the SCP-UPMA channel polling interval to 1 second and a mean message interval time from 2 to 10 seconds per node. We observe that YA-MAC’s message delivery ratio is much better than SCP-UPMA and is on a par with RI-MAC, as per our aim. Fig-7 shows where RI-MAC is superior in terms of throughput ratio to YA-MAC. This is due to RI-MAC’s hidden terminal contention resolution scheme (YA-MAC uses the default CSMA-UPMA framework). However, when the mean interval time for unicast traffic increases (representing slightly lighter traffic) we see that YA-MAC and RI-MAC have the same throughput ratio as shown in Fig-8.

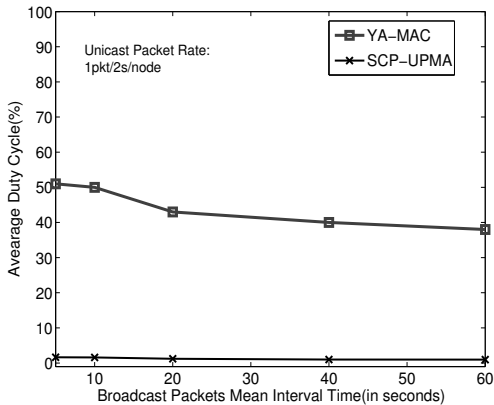
In terms of duty-cycles, YA-MAC is significantly better than RI-MAC by 40% and on a par with SCP as shown in Fig-7. This behaviour is also found when we experiment with lighter traffic where the mean message interval time goes from 20 to 120 seconds per node as shown in Fig-8. In terms of latency, SCP-MAC exhibits higher spikes above the average due to only

being able to deliver a single packet during a channel polling period, thus longer messages will take more than one period.

3) *SCP-MAC and YA-MAC with Mixed-cast messages*: In these experiments we mix unicast and broadcast messages. Here we keep the unicast message rate at 1 packet every 2 seconds per node for 47 nodes in Motelab. Broadcast messages are varied from 1 message every 1 second to 60 seconds. For YA-MAC we set  $\epsilon = 0.05$  with synchronicity threshold value at 60%. We expected that on mixing the messages we would get a performance hit for both broadcast and unicast messaging compared to their individual performances observed previously. We do observe a small decrease (5%) in throughput for unicast when the broadcast time interval is less than 40 seconds, however after this we achieve the same performance as before, shown in Fig-9(a) and Fig-9(b). More surprisingly, mixing the message types actually improved our broadcast message throughput by 15%. We believe this is because when the node awakens to unicast the sensor data, it has more opportunity to improve its *Synchronicity* value. Note that high duty-cycle of YA-MAC due to the large SETW,



(a) Average Delivery Ratio



(b) Average Duty Cycle

Fig. 9. Performance Comparison of YA-MAC and SCP-UPMA on a random network of 47 Telosb nodes, for mixed unicast and broadcast traffic where each node sends and receives 1 packet every 2 seconds as unicast traffic and the broadcast time interval ( $T$ ) varies from 1 to 60 seconds. For broadcast messages,  $\epsilon = 0.05$  and the *Synchronicity threshold* = 60%.

can be further improved by choosing the adaptive  $\epsilon$  which is explained in [14]. Fig-9(b) shows the worst-case duty-cycle where unicast traffic rate is high, i.e, 1 packet every 2 seconds per node. The duty-cycle keep decreasing with decrease in the unicast data rate and duty-cycle due to sender's wake-up time that can be further improved by predicting the precise wake-up patterns of the receiver nodes.

## VI. RELATED WORK

Energy saving has been given prominence in WSN MAC research and many duty-cycling techniques have been proposed and implemented to reduce energy consumption due to idle listening. By placing the nodes in sleep mode and coordinating duty-cycling in a synchronous fashion, energy consumption can be reduced [2], [11]. However when nodes wake-up at the same moment to communicate, collisions, overhearing, contention, etc., are increased. This has lead to a number of

improvements, such as synchronous TDMA MAC protocols that perform time slot-based node scheduling. Synchronous TDMA MAC protocols [3], [19] that solve collision, overhearing and idle listening problems to a great extent; but with added overheads to ensure time synchronisation. Therefore, these MACs are less efficient for dynamic topologies.

Asynchronous MAC protocols, on the other hand, provide flexibility to accommodate dynamic traffic without such overheads. Having said that, asynchronous MAC protocols, where nodes wake-up independently of synchronised schedules, also have their own costs. In asynchronous MAC protocols, either the sender has to initiate a request before transmission or a receiver has to initiate the request before receiving. Sender initiated asynchronous protocols [1], [12] send a preamble before the actual data which has a disadvantage that the channel is occupied for a longer duration reducing the network throughput. Therefore, this scheme is only useful for handling light traffic in the network. To improve throughput, optimised solutions have been proposed that provide better preamble handling [20], [21]. On the other hand, Receiver-initiated asynchronous protocols can efficiently handle dynamic unicast traffic in multi-hop networks, but fail to efficiently handle the (typically broadcast) control messages required by higher Routing and Application layers [4], [22], [23].

In order to mitigate some of the disadvantages of scheduled MACs, dynamic scheduling (such as in S-MAC [11]), bio-inspired mechanisms or other methods such as gradient based methods that evolve schedules or synchrony, have been introduced [24]–[26]. Strogatz and Mirollo's original mathematical model of firefly synchronisation [27] presented the configurations under which a fully connected pulse-coupled system can achieve synchronisation. Communications latency has the largest impact on this model's implementation in real networks, therefore the Reachback Firefly algorithm (RFA) [28] improved on this using information from the past to adjust their future firing phases. All these firefly based algorithms have high message overheads [13] and, to the best of our knowledge, the convergence of these algorithms in duty-cycled networks have yet to be fully explored.

YA-EBS extends MEMFIS by associating a refractory period (see RFA [28]) to its SETW but has shown to converge well in low duty-cycled networks [14]. In MEMFIS, a node updates its internal clock in a distributed manner when detecting a neighbouring transmission while keeping itself 100% duty-cycled to achieve the synchronicity between local neighbourhood. Also, like YA-MAC a small number of MACs have used traffic information to either adapt their duty-cycle or optimise other parameters [29]. Further, due to the application dependent nature of many WSN, the idea of cross-layer MAC designs have also been inevitably explored [30], [31].

However, after carrying out an extensive survey on duty-cycled MACs, we found that none of these solutions are able to handle both unicast and broadcast traffic while meeting the goals of optimizing duty-cycling, packet delivery ratios, scalability and stability, without significant time synchronisation overhead.

## VII. CONCLUSION AND FURTHER WORK

In this paper, we presented the design and evaluation of YA-MAC, which provides improved throughput for both broadcast and unicast traffic in dynamic DCM-WSN. We implemented YA-MAC under the UPMA framework in TinyOS showing significant improvements over RI-MAC in terms of lower energy consumption ( $\geq 35\%$ ) for unicast messages while achieving 60% more throughput than SCP-MACs for unicast messages. We also showed that under YA-MAC, mixed-cast messaging actually improves performance; achieving 99-100% packet delivery ratios under a steady state. However, YA-MAC has scope for further enhancement. The YA-EBS scheme currently expects homogeneous network wide values for broadcast time intervals,  $T$ , and *Synchronicity thresholds*. A study, incorporating different  $T$  values and *Synchronicity thresholds* for different nodes in the network, may lead to tighter optimisation to topologies of varying densities. Further, reducing 100% duty-cycling for the *broadcast slot emergence phase* could also bring our duty-cycle performance closer to SCP-MAC's.

## REFERENCES

- [1] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 95–107, ACM, 2004.
- [2] W. Ye, F. Silva, and J. Heidemann, "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), pp. 321–334, ACM, 2006.
- [3] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, "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), pp. 53–62, ACM, 2008.
- [4] Y. Sun, O. Gurewitz, and D. B. Johnson, "Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks," in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, (New York, NY, USA), pp. 1–14, ACM, 2008.
- [5] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links," in *MobiCom '09: Proceedings of the 15th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 133–144, ACM, 2009.
- [6] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza, "Rbp: robust broadcast propagation in wireless networks," in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 85–98, ACM, 2006.
- [7] X. R. Wang, J. T. Lizier, O. Obst, M. Prokopenko, and P. Wang, "Spatiotemporal anomaly detection in gas monitoring sensor networks," in *Wireless Sensor Networks* (R. Verdone, ed.), vol. 4913, pp. 90–105, Berlin, Heidelberg: Springer, Feb. 2008.
- [8] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. B. Johnson, "Adb: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks," in *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, (New York, NY, USA), pp. 43–56, ACM, 2009.
- [9] C. Sengul, M. J. Miller, and I. Gupta, "Adaptive probability-based broadcast forwarding in energy-saving sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, no. 2, pp. 1–32, 2008.
- [10] S. Ghosh, P. Veeraraghavan, S. Singh, and L. Zhang, "Performance of a wireless sensor network mac protocol with a global sleep schedule," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 4, 2009.
- [11] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *IEEE Infocom*, June 2002.
- [12] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "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*, (New York, NY, USA), pp. 307–320, ACM, 2006.
- [13] A. Tyrrell, G. Auer, and C. Bettstetter, "Emergent slot synchronization in wireless networks," vol. 9, (Piscataway, NJ, USA), pp. 719–732, IEEE Educational Activities Department, 2010.
- [14] P. Yadav and J. A. McCann, "Ebs: Decentralised slot synchronisation for broadcast messaging for low-power wireless embedded systems," tech. rep., [www.doc.ic.uk/research/technicalreports/2011/DTR11-4.pdf](http://www.doc.ic.uk/research/technicalreports/2011/DTR11-4.pdf). Imperial College London, 2011.
- [15] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *IPSN '09: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, (Washington, DC, USA), pp. 37–48, IEEE Computer Society, 2009.
- [16] K. Klues, G. Hackmann, O. Chipara, and C. Lu, "A component-based architecture for power-efficient media access control in wireless sensor networks," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 59–72, ACM, 2007.
- [17] *Harvard Sensor Network Testbed*: <http://motelab.eecs.harvard.edu/>.
- [18] *Indriya Testbed*: <http://indriya.ddns.comp.nus.edu.sg>.
- [19] J. Pak, J. Son, and K. Han, "A mac protocol using separate wakeup slots for sensor network," in *Computational Science and Its Applications (ICCSA '06)*, vol. 3981 of *Lecture Notes in Computer Science*, pp. 1159–1168, Springer Berlin / Heidelberg, 2006.
- [20] L. v. Hoesel and P. Havinga, "A lightweight medium access protocol (LMAC) for wireless sensor networks," (Tokyo, Japan), June 2004.
- [21] A. El-Hoiydi and J.-D. Decotignie, "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 Volume 2 (ISCC'04)*, (Washington, DC, USA), pp. 244–251, IEEE Computer Society, 2004.
- [22] H. Cao, K. Parker, and A. Arora, "O-mac: A receiver centric power management protocol," vol. 0, (Los Alamitos, CA, USA), pp. 311–320, IEEE Computer Society, 2006.
- [23] P. Yadav and J. A. McCann, "Qos based event delivery for disaster monitoring applications," in *Proceeding of the 5th International Conference on Wireless Communication and Sensor Networks(WCSN)*, 2009.
- [24] S. Suthaharan, A. Chawade, R. Jana, and J. Deng, "Energy efficient dna-based scheduling scheme for wireless sensor networks," in *WASA '09: Proceedings of the 4th International Conference on Wireless Algorithms, Systems, and Applications*, (Berlin, Heidelberg), pp. 459–468, Springer-Verlag, 2009.
- [25] S. Lai, B. Zhang, B. Ravindran, and H. Cho, "Cqs-pair: Cyclic quorum system pair for wakeup scheduling in wireless sensor networks," in *OPODIS '08: Proceedings of the 12th International Conference on Principles of Distributed Systems*, (Berlin, Heidelberg), pp. 295–310, Springer-Verlag, 2008.
- [26] A. Tyrrell and G. Auer, "Imposing a reference timing onto firefly synchronization in wireless networks," in *The 65th IEEE Vehicular Technology Conference(VTC'07)*, pp. 222–226, apr. 2007.
- [27] R. Mirolo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, pp. 1645–1662, 1990.
- [28] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 142–153, ACM, 2005.
- [29] R. L. Cigno, M. Nardelli, and M. Welzl, "Sesam: a semi-synchronous, energy savvy, application-aware mac," in *WONS'09: Proceedings of the Sixth international conference on Wireless On-Demand Network Systems and Services*, (Piscataway, NJ, USA), pp. 85–92, IEEE Press, 2009.
- [30] T. Canli, M. Hefaida, and A. Khokhar, "Bulkmac: a cross-layer based mac protocol for wireless sensor networks," in *IWCMC '10: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, (New York, NY, USA), pp. 442–446, ACM, 2010.
- [31] S. Du, A. Saha, and D. Johnson, "Rmac: A routing-enhanced duty-cycle mac protocol for wireless sensor networks," pp. 1478–1486, may. 2007.