

Self-Synchronization in Duty-cycled Internet of Things (IoT) Applications

Poonam Yadav *Member, IEEE*, Julie A. McCann *Member, IEEE*, Tiago Pereira

Abstract—In recent years, the networks of low-power devices have gained popularity. Typically these devices are wireless and interact to form large networks such as the Machine to Machine (M2M) networks, Internet of Things (IoT), Wearable Computing, and Wireless Sensor Networks. The collaboration among these devices is a key to achieving the full potential of these networks. A major problem in this field is to guarantee robust communication between elements while keeping the whole network energy efficient. In this paper, we introduce an extended and improved emergent broadcast slot (EBS) scheme, which facilitates collaboration for robust communication and is energy efficient. In the EBS, nodes communication unit remains in sleeping mode and are awake just to communicate. The EBS scheme is fully decentralized, that is, nodes coordinate their wake-up window in partially overlapped manner within each duty-cycle to avoid message collisions. We show the theoretical convergence behavior of the scheme, which is confirmed through real test-bed experimentation.

Index Terms—Internet of Things, Machine to Machine (M2M) Networks, Media Access Control (MAC), Radio Interference, Bio-inspired Algorithm, Firefly-based Self-synchronization, Broadcast messaging

I. INTRODUCTION

Networks of low-power wireless embedded systems play an important role in the successful adaptation of the heterogeneous components in Internet of Things and have gained wide attention because of their applications ranging from smart cars and cities to precision agriculture. The end-nodes (embedded devices) of these networks are resource constrained. For example, M2M or Sensor nodes make use of low-power radios such as Bluetooth [1] and Zigbee [2] for wireless communication. These low-power radios have a limited communication range, up to 100 meters, and data rate of a few hundred kilo-bits per second. Both design and maintenance of these networks are challenging [3], [4] because of two major constraints: battery-power-usage and lossy communication. Recent work has proposed a number of solutions for designing energy efficient and robust networks. One can consider a multi-hop networking to overcome the radio's low communication range, where a source node's data is delivered to the destination node through intermediate nodes. Moreover, duty-cycling can be used to optimize battery-power-usage and lengthen the network's lifetime [5], [6]. In duty-cycle networks, a node goes into a regular sleep mode to conserve the battery power. Combining both duty-cycled nodes

and Multi-hop (DCM) networking, one can obtain an energy efficient data delivery in a large size network. The challenge is that DCM networks bring its additional overheads mainly in two ways. First, an efficient time-synchronization scheme must be put in place to coordinate communication slot for the duty-cycled nodes. Secondly, DCM character reduces a node's overall network throughput which is sum of node's self-generated messages, forwarded messages and control messages. This occurs because a node has to forward messages to its neighbors along with its self-generated messages in only a short wake-up period. Moreover, the DCM character introduces further challenges due to the presence of unreliable wireless links and limited bandwidth. Hence, efficient route management mechanisms are central to DCM networks. Route management and DCM techniques aim to maximize network lifetime but at a cost. Route management mechanisms make use of control messages, which constitute extra traffic. Further, the node's duty cycling saves energy but also incurs management overheads to ensure nodes synchronize sleep-awake schedules. A number of *centralized* time synchronization algorithms have been proposed to achieve time coordination for low-power wireless networks [7]–[9]. They are unsuitable for DCM networks because of issues pertaining to maintenance overheads, synchronization delays, and single points of failure [10]. The potential application scenarios of DCM networks are represented by the industrial IoT applications such as moving robots in a warehouse [5], [11]–[13] or autonomous unmanned aerial vehicles (drones) [14] coordinate among themselves to achieve a common task. The wireless communication nodes on robots work in duty-cycled and multi-hop networking mode to save energy consumption. In this scenario, the Emergent Broadcast Slot (EBS) scheme [15] is suitable for achieving energy-efficiency as the scheme allows communication modules to stay in a full duty cycle to achieve synchronization among other nodes. Once such synchronization is attained, nodes go to sleep mode and wake-up only for a short time to transmit and receive information. During the wake-up time, nodes also update their clocks, which keeps the synchronization stable and communication robust. If the quality of synchronization is compromised, the nodes then return the full duty cycle to recover synchronization. This flexibility, with no overheads, saves energy and leads to a robust scheme able to cope with dynamic changes.

In this paper, we present an extended version of Emergent Broadcast Slot (EBS) scheme [15] that provides an energy efficient and robust communication. In our scheme the devices stay in sleep mode and wake up in a synchronous manner for communication. This synchronization is spontaneous

P. Yadav is with Computer Laboratory, University of Cambridge and J. A. McCann is with the Department of Computing, Imperial College London and T. Pereira is with the Institute of Mathematics and Computer Science, University of Sao Paulo, Brazil, e-mail: (poonam.yadav07@alumni.imperial.ac.uk).

and requires minimal overheads. Our synchronization protocol is biologically inspired by the firefly model and combines recent developments in the biologically inspired synchronization algorithms. Our extensive experimental and theoretical analysis reveals that the EBS has following two characteristics:

- *Quick Convergence and Robustness*: EBS converges exponentially fast in a fixed network structure. Moreover, in a mobile and ad-hoc network where nodes leave or join or move within the network, the scheme is robust and readapt at no significant performance cost.
- *Energy and Transmission Efficiency*: While keeping nodes awake only 5% of their duty cycle the scheme provides 95% effective message transmission. This saves energy and avoids message collision.

The paper is organized as follows: In Section II, we present related work on decentralized time synchronization schemes and firefly-based synchronization approaches. In Section III, we present an overview of EBS. In Section IV, we present the EBS *synchronization state* dynamics. EBS configuration parameters and their effects are analyzed in Section V. We present two implementations of EBS in Section VI and discuss their suitability and convergence in the presence of random delays. We briefly discuss our test-bed experimental setup in Section VII, and the resulting comparative performance results are presented in Section VII. We then conclude the summary along with future work in Section VIII.

II. RELATED WORK: SYNCHRONIZATION SCHEMES

In general, low-power nodes consist of relatively inexpensive hardware components, e.g. clocks that typically drift. Therefore, network-wide synchronization is required, thereby increasing message overheads as well as temporal and spatial instability in the network [16], [17]. Using a single central device to enforcing synchronization by dictating the time is not ideal as it imposes extra messages overload [8], [18]. Moreover, time delay for the synchronization message keeps incrementing with every hop. An additional challenge is the presence of lossy wireless links means that the synchronization packets may be dropped and would have to be sent multiple times.

Both distributed coordination and local synchronization are used to increase stability and facilitate broadcast message transfer [19]–[21]. Examples of such schemes include gradient-based [22] and bio-inspired algorithms [23]. Typically bio-inspired algorithms have higher overheads than gradient-based algorithms [24]–[26]. However, Gradient-based schemes can be ridged and unable to cope with the dynamism such as node failures [27]. On the other hand, the emergent nature of firefly-based synchronization can cope with failure. Additionally, when a failure happens in another part of the network firefly-based synchronization does not affect the local cluster.

A. Background on Firefly Based Synchronization

The pulse-coupled oscillators (PCO) model is used to capture the synchronization behavior observed in fireflies flashing in unison [28]. In this model, each firefly is described as a periodic oscillator running at its natural frequency – the frequency defines the number of times the firefly flashes in a unit time. And, the period defines the time interval between two consecutive flashes.

The state of a periodic oscillator can be described by its phase $\phi(t)$ evolving over time t from 0 to 1. The firefly *fires or flashes* when its phase $\phi(t) = 1$ and after that resets again to $\phi(t) = 0$. To achieve synchronization, the oscillators in the system have to interact by sending and receiving pulse signals. When a PCO's phase $\phi(t) = 1$, it emits the pulse signal (fire event) to interact with the neighbour PCOs. Upon reception of the pulse signal, each neighbour PCO advances its phase $\phi(t)$ to a new phase $\phi(t^+)$ by taking a phase jump $\Delta\phi(t)$, which indicates the next time it will fire and is calculated by using a *phase advancement function*, see Figure 1 for an illustration. The different firefly based synchronization algorithms make use of different phase advancement functions designed in such a way that if all the oscillators follow a given function, they all will synchronize eventually.

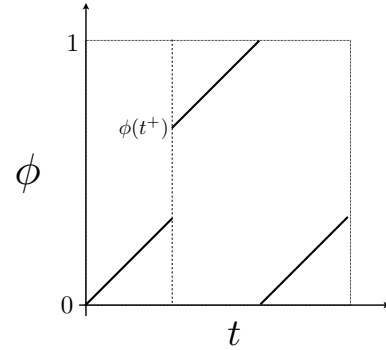


Fig. 1: Phase Dynamics of a Pulse-Coupled Oscillator (PCO). We illustrate its behavior by showing the relationship between its phase versus time. The PCO phase increases linear with time, however, when the neighbor node fires (illustrated by the dashed line) its phase is updated to a new phase value $\phi(t^+)$. When the phase reaches 1 it is reset to 0.

B. Related Work Based on Firefly Based Synchronization

Because in the PCO synchronization is an emergent property, this model has attracted much attention and has been adapted to wireless systems. For instance, variations of the model to include synchronization *frustration* [29] and inhibitory pulse coupling [30] have been introduced. An adaptive Ermentrout Model based synchronization scheme is proposed in which frequency of the flashing is adjusted instead of phase when a node receives a flash from another node [31]. Other relevant work includes the *Reach-back* Firefly Algorithm (RFA) that accounts for communication latencies by allowing nodes to use neighbours' firing information from the past to adjust the future firing phase [23]. However, due to propagation and processing delays in the notification of the firing-event,

nodes keep firing which leads to chasing conditions. To avoid this, Degesys et al. [32] introduced the concept of *refractory periods*. The *refractory periods* are time periods that start just after node fires, and during this time, the firing node ignores fire messages from other nodes to avoid the aforementioned chasing behavior. If the network size and connectivity are moderate the refractory period has negligible influence on the probability of the synchronization [33] [34].

Another extension of the previous protocols is the extended Reachback Firefly Algorithm (e-RFA) for wireless sensor networks [35]. The synchronization algorithm uses both the *Refractory Period* and the *Reachback* concepts and rate calibration scheme for longer re-synchronization intervals. The algorithm is evaluated using simulations and test-bed in a mesh network topology. However, algorithm performance is presented for random multi-hop network and as well as in the presence of non-deterministic propagation delays. Implementing the RFA in conjunction with a Late Sensitivity Window, in which nodes do not advance their phase, not only improve the percentage of synchronized nodes, but also reduces the time to synchronize and the number of broadcast messages [36].

To reduce the number of broadcast messages and the time to synchronize an Meshed Emergent Firefly Synchronization (MEMFIS) has been proposed [10]. This scheme relies on the detection of a synchronization indicator (text/symbol) common to all nodes. This indicator is embedded into each packet with payload data. As MEMFIS embeds synchronization indicator in every payload message, it does not require separate synchronization messages but only uses payload messages as synchronization messages, which is an advantage. But embedding the synchronization indicator in every payload message also becomes a limitation when the number of network payload messages are significantly high because synchronization information consumes a significant portion of network bandwidth, lowering throughput.

Pagliari et al. [37] implemented a scheme where an additional radio stack that is used for only synchronization messages, whereas regular CSMA radio stack for the data load. This scheme achieves fast synchronization as compared to the CSMA based schemes. However, this scheme is complex as it implements a separation of a duty-cycle into two parts for each radio stack at hardware level, which introduces more synchronization messages collisions.

Yadav et al. [38], [39] presented a receiver initiated medium access control protocol, which uses bio-inspired scheme (the initial version of EBS scheme [15]) for route discovery messages. The paper demonstrated the extended scope of the EBS scheme in the networking stack of the wireless sensor networks and low-power wireless networks.

In summary, bio-inspired and decentralized synchronization schemes have shown great promise. However, to be used in real-world situations, they have to be able to work with duty-cycled systems. All current approaches ignore the fact that re-synchronization is required after a sleep period, and they also ignore the effects of asymmetric wireless links in dense networks; i.e., they are intolerant to the missing synchronization messages. Further, none of the aforementioned related schemes takes energy efficiency into account for DCM networks.

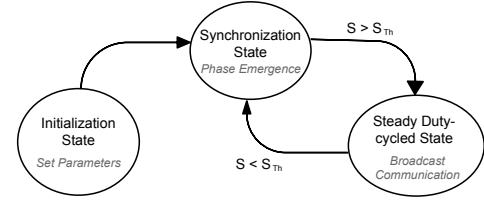


Fig. 2: A node is awake during the Initialization and Synchronization states, and in the Duty-cycled state, it wakes-up during its SETW.

Our previous contribution [38] presented an initial version of the EBS scheme showing the behavior of the scheme using the selective values of the configuration parameters. However, it was unclear why some variable settings led to unstable behavior [38]. In this paper we evaluate the EBS scheme using extensive simulations and theory, thereby deriving the relationship among different parameters and validating the EBS performance on a real testbed.

III. EMERGENT BROADCAST SLOT (EBS) SCHEME

In a typical duty-cycled multi-hop network, each node exchanges broadcast messages with its one hop neighborhood nodes (i.e., the nodes in its direct communication range) periodically [40]–[42]. To minimize energy consumption, each node should remain awake for a small wake-up window within every periodic time interval T . Our EBS scheme provides an algorithm to achieve this small duty cycle condition.

In the EBS Scheme, each node in the network will be in one of three states: the initialization state, synchronization state, and the steady duty-cycled state. Roughly speaking, in the initialization state the nodes are initialized and then transition to a synchronization where they will synchronize their wake-up windows; once this is achieved the nodes can go to sleep mode and only awake during the window to exchange messages. This last state is the steady duty-cycle. One important aspect of the scheme is that if the nodes feel that synchronization is compromised either by node failure or by broken connection, then they transition back to the synchronization state. In Figure 2 we present a schematic representation of the stages.

A. Terms and Definitions

Since each node is periodic (a clock) we can describe its state in terms of a phase variable $\phi \in [0, 1]$. Recall that the node broadcast its message when ϕ equals 1. We define

$$\phi'(t) = 1 - \phi(t)$$

as the remaining phase left, after which a node is scheduled to broadcast its next control message. See Figure 3 for an illustration of the phase dynamics.

The EBS scheme makes use of the following:

Synchronization Error Tolerance Window (SETW) Given a time window T and a synchronization error tolerance $\varepsilon > 0$, we define $SETW = [-\varepsilon T, \varepsilon T]$, see Figure 3

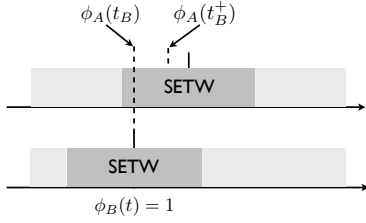


Fig. 3: Phase dynamics in the synchronization state when the node is awake for the whole period. This figure shows the phases ϕ and ϕ' of the node A when it receives a broadcast message from B at time t_B .

for an illustration in terms of the phases. During the duty-cycle phase the nodes will only awake during their SETW.

Synchronicity (S) Let N_i be the set of one-hop neighbors of the node s_i . Moreover, let H_i be the one-hop neighbors that the node s_i can hear in its SETW. We define the synchronicity S_i of node s_i as:

$$S_i = \frac{|H_i|}{|N_i|} \times 100$$

where $|A|$ denotes the number of elements of the set A .

Synchronization Threshold (S_{Th}) A node's S_{Th} provides the minimum percentage of nodes from its neighborhood that it must hear while it is awake; i.e. during its SETW. Therefore, S_{Th} will control the synchronization level we wish to enforce.

B. Initialization State

In the initialization state, all nodes are awake for the whole period, i.e., 100% duty cycled. In this state the nodes start the random timer and initialize parameters such as the broadcast time interval T . Next, the nodes calculate their neighborhood size. This state lasts only a few cycles and then transitions to the *synchronization state*.

C. Synchronization State

Each node coordinate its SETW with its neighbors' using a PCO model. The node remains awake until their slots converge and then they sleep only to awake during the SETW. A node declares itself synchronized¹ if its *Synchronicity*, S is above or equals to S_{Th} . Once a node declares itself as synchronized, it transitions to the steady duty-cycled state.

To achieve the required S we use a PCO model with a particular phase advancement function

$$\phi(t^+) = \begin{cases} 1 - g(\phi(t)) & \text{if } \varepsilon < \phi(t) < (1 - \varepsilon) \\ \phi(t) & \text{Otherwise,} \end{cases} \quad (1)$$

with $\varepsilon \leq (0.5)$, and $\phi(t^+)$ denotes the new phase value and $\phi(t)$ is the previous phase value. g is the phase advancement function given by:

$$g(\phi(t)) = \sigma(1 - \phi(t)) \quad \text{with } \sigma \in (0, 1).$$

¹For scalability the synchronization decision is local to every node.

Suppose, node A receives a broadcast message from node B at time t_B as shown in Figure 3. Then node A checks the time it has passed since its previous broadcast message in the current period, represented by $\phi_A(t_B)T$. According to the Equation (1), if $(\varepsilon < \phi_A(t_B) < (1 - \varepsilon))$, it shortens its remaining phase $\phi'_A(t_B)$ to $g(\phi_A(t_B))$.

In ideal conditions, to achieve synchronization fast, the node transmits its broadcast message immediately by setting $\phi_A(t_B^+) = 1$ and $g(\phi_A(t_B)) = 0$. After transmission it resets the next broadcast message transmission time to T and $\phi_A(t_B) = 0$. However, recall that all nodes within the one-hop neighborhood can listen to the same broadcast message and if they were to broadcast their messages on receipt of this broadcast message simultaneously, then there will be many messages in the local area; leading to packet collisions². To avoid such collisions, nodes delay their broadcast message transmission by $(\phi'(t^+))$ which is equal to $g(\phi(t))$ without disturbing the synchronization.

The appropriate values of ε and σ depend on a number of factors such as the number of nodes in a node's 1-hop and 2-hop neighborhoods, the value of T , as well as message propagation delays (discussed further in Section V). The details of how synchronization is achieved in this state are discussed in the next Section IV.

D. Steady Duty-cycled State

Once synchronized, the nodes switch to the *duty-cycle phase* and wake-up only during their respective SETW to exchange messages. In the *duty-cycle state*, all nodes also constantly update their S values. The nodes whose S values fall below the S_{Th} , switch back to the *Synchronization State* to recalculate the S_{Th} for a single T period by returning to 100% duty-cycling and then switching to the *Synchronization State*. The various reasons for a drop in a node's S value can be due to clock drift, change in network density, etc. The two common cases are:

New node joins the network: It is the new node's responsibility to synchronize itself with the already synchronized network. First, it follows two first states gathering information regarding the number of neighbors it can hear, and then synchronizes its SETW with its neighbors. To this end, the node calculates S and it switches to the *Steady Duty-cycled State* if its $S \geq S_{Th}$. If the new joining node is mobile and already in the *Steady Duty-cycled State*, then it switches to the *Synchronization State* only if $S \leq S_{Th}$. The neighbor nodes, who are now synchronized with this node, find their neighbors count within SETW is incremented by 1. Until synchronicity S of the neighbor nodes satisfy the condition $100 \geq S \geq S_{Th}$, they continue without any change, but if $S > 100$, then nodes update their neighbors count and S_{Th} .

A node leaves the network: When a synchronized node leaves the network, it decreases the value of S of its neighboring nodes as well as total number of neighbors. If this

²Original CSMA back-off mechanism introduces delays greater than the SETW which disturbs synchronization that as a result increases message losses. Therefore, in our implementation we keep CSMA back-off and re-transmissions to the minimum value to keep synchronization less affected, which as a result makes CSMA more effective.

change drops their value in such way that $S < S_{Th}$, then nodes update their number of neighbors and S by switching back to *Synchronization State* otherwise they continue without any change, however, new neighbor count will be updated only when nodes go to the *Synchronization State*.

This method of switching back has the advantage that, unlike gradient based schemes [22], one node's disturbance does not perturb the whole network; therefore, EBS scheme is both agile and tolerant of change. The EBS requires the node to wake-up only once within the period T rather than multiple times. Recall that multiple wake-up periods within a T , can cause higher-energy consumption in terms of radio state transitions as the transceiver is required to power up, transmit and receive messages for each period. All current distributed or centralized slot-based time synchronization schemes used in multi-hop networks expect to have multiple wake-up periods [43], [44].

IV. EBS SYNCHRONIZATION STATE DYNAMICS

We model the EBS System as a complex dynamical network represented by a graph $G(D, L)$, that is composed of n nodes represented by the set $D = (s_1, s_2, \dots, s_n)$ where L is a set of all transmission links between nodes in D . The dynamics of each node s_i in the network is characterized by the phase ϕ_i . Moreover, the one-hop neighbourhood of a node s_i is represented by the set N_i .

We start by examining the EBS dynamics in the absence of transmission delays. EBS is required to achieve synchronization to a given precision to avoid the message collisions that would be present³. Our goal is to study the ability of the EBS to quickly synchronize so that the number of nodes in 100% duty-cycle states are minimized. To this end, we make use of two metrics: the average phase difference and the average phase advancement. The average phase difference is given by:

$$\Delta\phi = \frac{1}{n} \sum_{s_i \in D} \frac{1}{|N_i|} \sum_{s_j \in N_i} |\phi_i - \phi_j|. \quad (2)$$

Here, N_i represents the neighbours of node s_i and the total number of neighbours is $|N_i|$.

The average phase advancement at time t in time-period T is calculated as follows:

$$\Delta^+ = \frac{1}{n} \sum_{s_i \in D} |\phi_i(t^+) - \phi_i(t)|. \quad (3)$$

The Δ^+ provides the average phase displacement a node must perform. So, decreasing Δ^+ leads to a better performance of the scheme and while achieving the required precision.

A. Synchronization Error-Tolerance

We wish to derive a stability condition for pairwise synchronization [45], [46]. Consider a given node s_i and let s_j belong to N_i that one-hop neighbourhood of s_i . Next, we assume

³There is a trade off between the SETW size and the probability of network collisions. A smaller SETW has energy saving advantages but leaves a smaller time interval within which messages can be communicated thus increasing collisions.

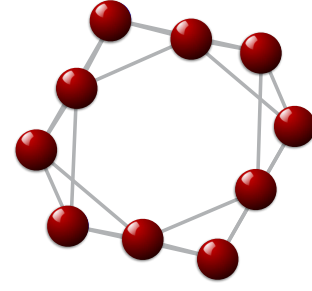


Fig. 4: Simulation Topology

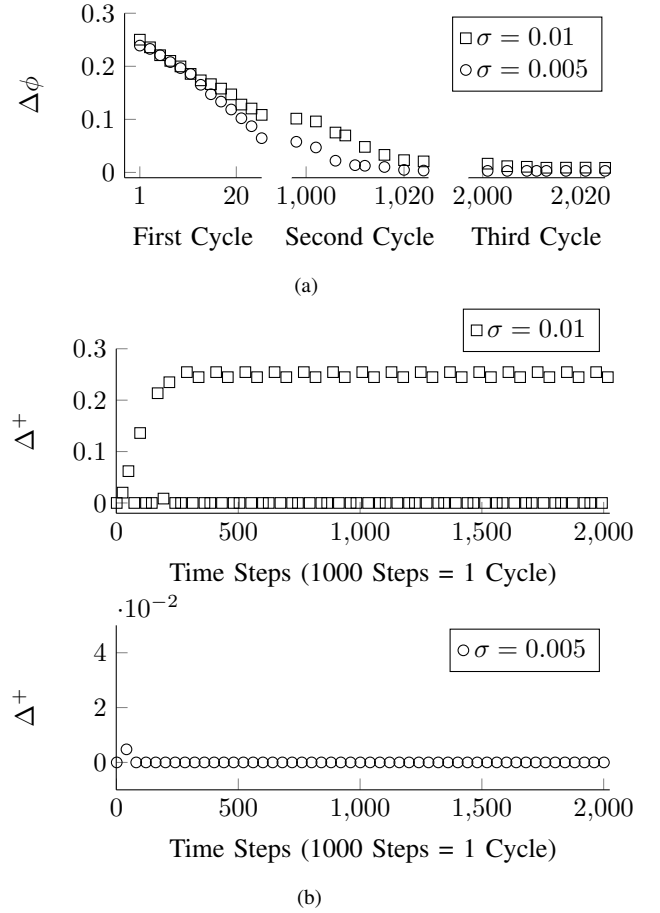


Fig. 5: Convergence behavior of EBS with the different values of σ when $\delta = 0$ and $\varepsilon = 0.01$. Fig (a) shows that the average phase-difference converges to zero for different values of σ after few cycles. Fig (b) shows that average phase advancement converges to zero when $\sigma = 0.005$ and stays to a constant value when $\sigma = 0.01$, hence represents non-convergence. It is clear from the figures that EBS achieves both required precision and stability when values of σ and ε satisfy Equation (4).

that the two nodes are synchronized, that is, they fire within the SETW. The synchronization condition imposes that when the node s_i fires at time t the new value $\phi_j(t^+)$ must be in SETW. This translates to :

$$|\phi_i(t) - \phi_j(t^+)| < \varepsilon,$$

using Equation (1) we obtain the fact that when s_i fires we have $\phi_i(t) = 1$,

$$|\sigma(1 - \phi_j)| < \varepsilon,$$

notice that $|1 - \phi| \leq 1 - \varepsilon$, hence the stability condition reads as

$$\sigma < \frac{\varepsilon}{1 - \varepsilon}. \quad (4)$$

We analyzed the convergence behavior of EBS through simulations using different network topologies. However, due to space limit we present convergence behavior for a regular network with four nearest neighbours as shown in Figure 4, because it provides a good example of complex scenario where each node has four connections with triangle cliques.

We first analyzed the behavior of EBS for different values of ε and σ in the absence of delay. In Figure 5 (a), we found that for different values of σ , EBS reaches the required average phase difference (refer Equation (2)) in two cycles. To better understand how well the duty cycle is being achieved we also analyzed the average phase advancement (refer Equation (3)) of the nodes over time. If the value of this metric is greater than 0, then the nodes phase-jump using the EBS phase advancement function. In Figure 5 (b), we see that the average phase advancement value Δ^+ in every time period T reduces to 0 after a few time steps when our stability criterion shown in Equation (4) is met.

B. Convergence in Presence of Deterministic Delays

Let the phase δ represents a phase change during message propagation between any two directly connected nodes. Notice that δ represents a deterministic delay. In this case, to obtain synchronization between the SETW's we require:

$$|\phi_i(t) - \phi_j(t^+)| < \varepsilon - 2\delta. \quad (5)$$

When all the delays are deterministic and satisfy the above Equation (5), then the system converges within the given precision bounds as shown in Figure 6. In Figure 6 (a) we present the convergence behavior of EBS for various values of δ when $\varepsilon = 0.1$ and $\sigma = 0.01$. We observe that, as the value of δ increases, the average phase difference $\Delta\phi$ also increases and meets the boundary condition shown in Equation (5). The Figures 6 (b) shows the value of average phase advancement that converges to zero when $\delta = 0.0$ and 0.01 , which represents EBS convergence whereas represents EBS non-convergence behavior when the average phase advancement value keeps fluctuating between 0 and 0.04 at $\delta = 0.1$ and 0.05 .

Our numerical experiments also suggest that $\varepsilon > 2\delta$ is required to achieve stability. Moreover, if the stability criterion is satisfied, higher values of σ leads an improvement of the precision of synchronization. Hence, we study and derive the relationship between σ , δ and ε in the next Section V.

V. EBS CONFIGURATION PARAMETERS

The performance of the EBS scheme depends on the values of its three configuration parameters: ε , σ and S_{Th} . These parameter values reflect the network density and the

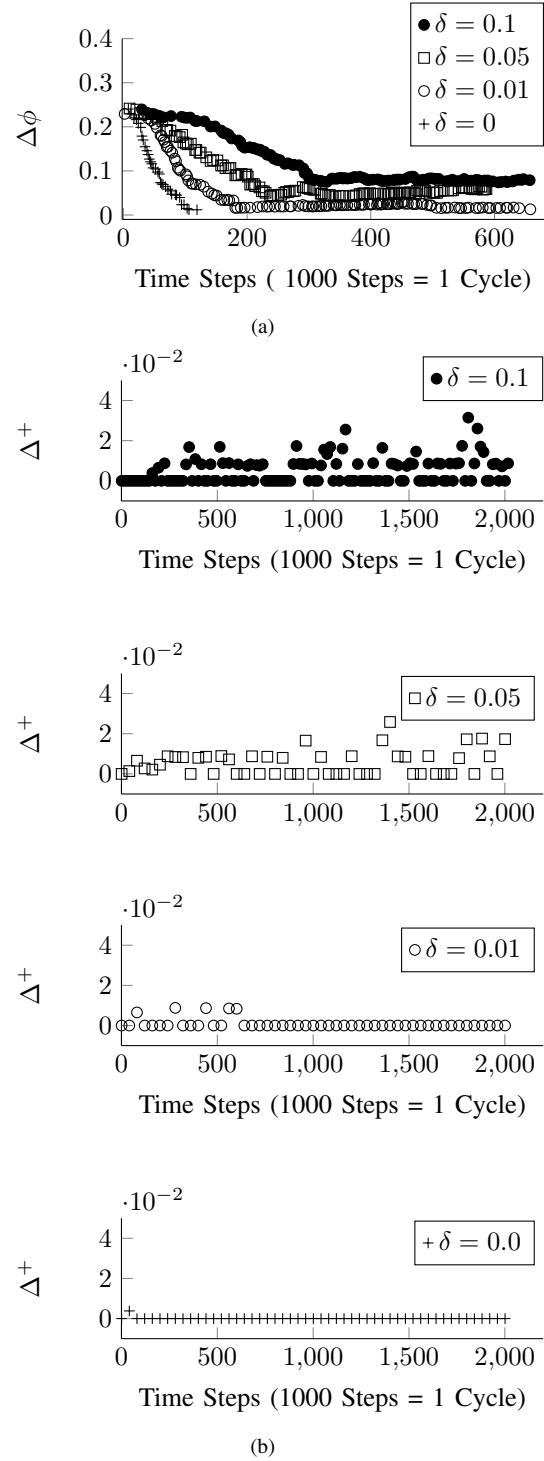


Fig. 6: Convergence behavior of EBS with different values of δ when $\sigma = 0.01$ when $\varepsilon = 0.1$. Fig (a) shows that value of δ increases, the average phase difference $\Delta\phi$ also increases and meets the boundary condition shown in Equation (5). Fig (b) shows that average phase advancement converges to zero when $\delta = 0.0$ and 0.01 (represents EBS convergence) and stays to a fluctuating value when $\delta = 0.1$ and 0.05 , hence represents EBS non-convergence.

uniformity of the nodes in terms of degree of connectivity in the network. For a given average degree connectivity per node in the network, we analyze ε , σ , and S_{Th} , respectively.

Effects of ε – The ε parameter plays an important role in the EBS scheme because it is directly proportional to the duty-cycle, that is, the awake period of a node corresponding to its SETW. In a DCM network, we wish to keep energy consumption of a node to minimum by maintaining a small value of ε and SETW.

However, regarding the convergence of the EBS scheme, the size of the SETW plays an inverse role, that is, the larger the ε and SETW, the quicker the EBS scheme converges. Therefore, a trade-off is required to find the minimum size of SETW that can guarantee fast convergence without the problem of overshooting as shown in Figure 7.

The overshooting situation occurs when node A receives a broadcast message from node B after a time-delay ($\nu_{B \rightarrow A}$), see Figure 7. This overshooting situation leads to a chasing condition, both nodes will keep on updating their phases. To avoid this overshooting condition, we can limit the lower bound of the εT to:

$$\varepsilon T \geq (\nu_{A \rightarrow B}) + g(\phi(t_B))T + (\nu_{B \rightarrow A}). \quad (6)$$

If $\nu_{A \rightarrow B} \approx \nu_{B \rightarrow A} = \nu$, Equation (6) can be rewritten as: $\varepsilon \geq g(\phi(t_B))T + 2\nu)/T$. Notice that in this construction $\varepsilon \leq 0.5$. Therefore, we obtain:

$$0.5 \geq \varepsilon \geq g(\phi(t_B)) + \frac{2\nu}{T}. \quad (7)$$

So, even if the nodes are synchronized $g(\phi(t_B)) = 0$ (when we consider the immediate transmission of broadcast messages), the lower value of ε is bounded by $\frac{2\nu}{T} \approx 2\delta$. The optimal value of ε is chosen by considering the duty-cycle requirements and density of the network.

For example, we let β be the maximum time a node needs to receive and process a broadcast message. Assuming that each neighbourhood node sends a single message in T . So, the maximum time a node is required to be awake to receive and process messages from its neighbourhood of size $|N|$ is $\beta|N|$; thus the size of SETW should be $|SETW| = \beta|N| + 4\nu$, which yields $2\varepsilon T \geq \beta|N| + 4\nu$. So, the optimal value of ε reads as:

$$\varepsilon_{opt} = \frac{\beta|N| + 4\nu}{2T}. \quad (8)$$

Effects of phase advancement function – Our coupling function is linear and σ defines the rate of convergence in the *initialization phase*. Recall that a large value of σ slows down the convergence of the EBS scheme. For faster synchronization, we must choose smaller values of σ . Notice that we do not use $\sigma = 0$, because this condition leads to message collisions in the network⁴. The maximum value of the coupling σ can be obtained by performing a similar analysis

as in Equation (4). In the presence of transmission delays we obtain:

$$\sigma_{max} \leq \frac{\varepsilon - \frac{2\nu}{T}}{(1 - \varepsilon)}. \quad (9)$$

VI. EBS TEST-BED IMPLEMENTATION DETAILS

To observe real-world behaviors, we implemented the EBS scheme on a test-bed. We used the UPMA [47] framework in TinyOS 2.x for the CC2420 radio, compliant to *IEEE 802.15.4* standards, with a data rate of 250 kbps. The UPMA framework is built on the CSMA MAC, and we chose to couple the EBS components to the MAC layer instead of the application layer to avoid buffer delays caused by the intermediate stack queues and buffers. We implemented two versions of the EBS scheme to understand its behavior. The two implementations are: (a) EBS without Reach-back⁵ to enable stand-alone synchronization support in the absence of upper-layer broadcast messages, and (b) EBS with partial Reach-back in which the broadcast messages are those generated from an application or routing layer, and are piggybacked with a sync-byte representing the EBS broadcast messages.

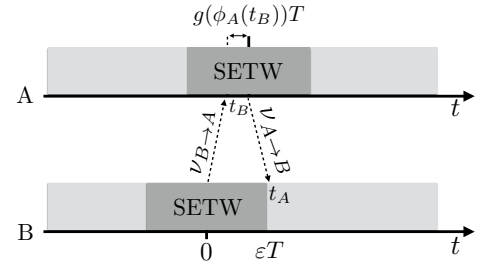


Fig. 7: Overshooting condition occurs when node B receives broadcast message from node A at its time t_A , which is out of its SETW.

A. EBS without Reach-back

In the latter implementation of the EBS scheme, when a node receives the broadcast-message from other nodes, it advances its phase according to the EBS phase advancement function Equation (1). The node broadcast the messages each time when $\phi(t) = 1$. This means, if there is a message pending to be transmitted at $\phi(t) = 1$, EBS piggybacks this message with its sync-byte and transmits it as the EBS broadcast-message. If there are no pending messages at the time, then the EBS layer creates a new synchronization message and broadcast it.

The advantage of this scheme is that nodes converge quickly in the *synchronization state* by exchanging messages without

⁴A node immediately transmits its broadcast message, and if all the nodes in the local neighbourhood do the same, it leads to message collisions.

⁵Recall reach-back allows a node to keep a record of the phase advancement of the current period and advances its phase immediately at the start of the next time period instead of firing immediately at current period.

the need for *reach-back*. However, this scheme has a high cost when overshooting occurs in *synchronization state*. The overshooting occurs when $\varepsilon < 2\delta$ or $\varepsilon < (\nu_{A \rightarrow B} + \nu_{B \rightarrow A})$, as previously derived in Section IV-B. Here, the nodes chase each other's broadcast messages without achieving stable synchronization. As a result, the whole network becomes congested with EBS broadcast messages, the network may never go to *synchronization state* irrespective of S_{Th} . If overshooting occurs temporarily in *steady duty-cycled state*, for lower S_{Th} , the network flips back to the *synchronization state*.

B. EBS with Partial Reach-back

The EBS scheme is implemented at the MAC layer; the EBS layer receives a broadcast message from the application or routing layer and transmits it after piggy-backing as EBS broadcast message. To establish coordinated SETWs during the *synchronization state*, a node advances its phase but transmits broadcast messages only when it has a new schedule broadcast message. The advantage of this scheme is that when overshooting occurs this scheme avoids the network flooding with broadcast messages, unlike the above implementation. However, this leads to nodes state flapping between *synchronization* and the *duty-cycled states*, shown experimentally in Figure 8. In Figure 8, $T = 10$ s and we

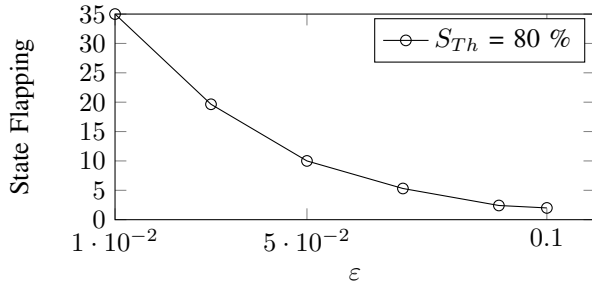


Fig. 8: Individual behavior of a node, which has average connectivity 20 in Motelab Test-bed, where all nodes running EBS with reach-back.

fixed $\varepsilon = 0.01$ that is smaller than the $\varepsilon_{opt} \approx 0.04$ derived from Equation (8), and a node flaps between synchronized and non-synchronized modes. That is, when a node is in its *synchronization state*, it switches to a *duty-cycle state*, however, in the non-synchronized mode, it returns to the 100% duty-cycled mode again. Further, when $\varepsilon = 0.05$, then the flapping reduces significantly as predicted by our theory.

C. Neighbourhood Size

For each node to understand its degree of synchronicity, the EBS implementation requires information regarding the neighbourhood size to calculate a node's S_{Th} . One approach is to get the neighbourhood size at the start by keeping all nodes awake for the initial few periods of T and identifying its neighbours via overhearing. To make the EBS scheme more agile, it is better to update neighbourhood size periodically during the EBS *duty-cycled state*. Nevertheless, in the implementation presented in this paper, EBS does not maintain its own neighbourhood table but uses the neighbourhood size, calculated

from the number of broadcast messages overheard in a single broadcast message interval T while it was in the *initialization state*. Alternatively, link estimation mechanisms found in many routing protocols require neighbourhood information so this can be freely used to make a precise neighbourhood size estimation.

VII. EXPERIMENTAL SETUP AND RESULTS

We perform our experiments on two different size test-beds: Motelab (87 Telosb nodes) [48] and a smaller 10 nodes MicaZ testbed [2]. We begin our experiments by fixing the value of the broadcast time interval T and varying the synchronization threshold S_{Th} , then we measure the duty-cycle of the network after every second interval. We present the results of the experiments executed on Motelab, with a broadcast interval time T set at 10, 20, and 30 seconds (representing typical routing control message intervals). Recall the SETW width varies according to the Equation (8).

For each run (either varying T or S_{Th}), we measured the duty-cycle of each node and their wake-up times, as a percentage. In the resulting graphs, we present the average duty-cycle and average throughput of the network by varying the S_{Th} from 95 to 20. The average duty cycle is calculated by keeping a internal timer for each node, which keeps record of wake-up slots during the run time of the experiment. We then use average duty cycle of each node to calculate the network-wide average duty cycle. For broadcast communication, we use the following Equation to calculate the throughput percentage in one time period:

$$\text{Throughput} = \frac{\text{Sum of all broadcast messages received}}{\text{network average degree} \times n} \times 100$$

The average network-wide degree of connectivity is calculated by sum of all the messages received divided by total number of nodes in the 100% duty-cycled mode. The EBS scheme exchanges broadcast messages in SETW, for unicast messages, it can easily be integrated with unicast message handling MAC layer [38].

In the next Section, we present EBS performance results showing both convergence parameter effects and performance and compare with the nearest state-of-the-art slot management protocol.

A. Calculation of Deterministic Propagation Delay

From Equation (8), it is clear that the value of ε_{opt} is proportional to propagation and processing delays β . The value of β is non-deterministic in real environments, however the minimum value ($C_0 \leq \beta$) as a constant parameter that is required in EBS scheme to calculate size of SETW to ensure convergence in the *synchronization state*. To get approximate value of C_0 , we first performed our initial experiments to observe the behavior of the EBS with a prefixed ε varying from 0.01 to 0.1. We found that nodes flapping with a low value of ε are more prominent compared to large values of ε , shown in Figure 8.

We ran initial experiments to derive the value of C_0 , we found it to be $C_0 \geq 50$ ms for the network to converge when

the average connectivity of this network was 2–3 nodes. However, in a non-uniform network, where some nodes have high degree of connectivity, the EBS calculates the adaptive value C^i of node i that is necessary for achieving synchronization using constant value C_0 and the node's neighbourhood density and other factors that can be further explored. However, for the experiments in this paper, we choose to implement EBS (without *reach-back* implementation) with adaptive C^i given as follows:

$$\begin{aligned} C^i &= (C_0 \times |N_i| \times S_{Th}/100) \text{ ms} \\ &= (50 \times |N_i| \times S_{Th}/100) \text{ ms} \end{aligned} \quad (10)$$

Note that the initial experiments shown in Figure 8, are performed taking into account $\sigma_{max} \leq \frac{\varepsilon}{(1-\varepsilon)}$ and by assuming $2\nu/T = 0$ in Equation (9).

B. Effects of Synchronization Threshold S_{Th}

We study the effect of the Synchronization Threshold on the nodes' convergence in the *synchronization state*. For these experiments, we set $T = 30$ s and vary the synchronization threshold S_{Th} from 20 to 95, the results are shown in Figure 9.

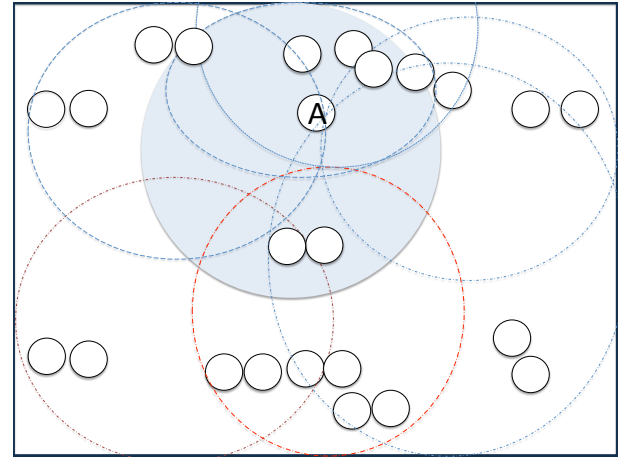
Figure 9 (a) shows the Motelab [48] topology on which we ran our experiments. Figure 9 (b) is composed of two figures, showing the comparative duty-cycle (%) and throughputs (%) achieved.

The figures present an average duty-cycle and throughput with $C = C_0 = 50$ ms for all nodes, node A's average duty cycle and throughput with $C = C_0 = 50$ ms, and average duty-cycle and throughput with adaptive C which is calculated by each node using Equation (10) of all nodes in the given topology. Due to high degree of connectivity, Node A goes out of synchronization for high value of $S_{Th} \geq 80$ when $C = C_0 = 50$ ms, which results in decreased throughput as shown in Figure 9 (b). To avoid these situations, adaptive C is proposed. We found that adaptive C follows a regular pattern regarding both duty-cycle and throughput results. Furthermore, adaptive C improves the throughput significantly by 40% at the small cost to the duty-cycle (2–3%).

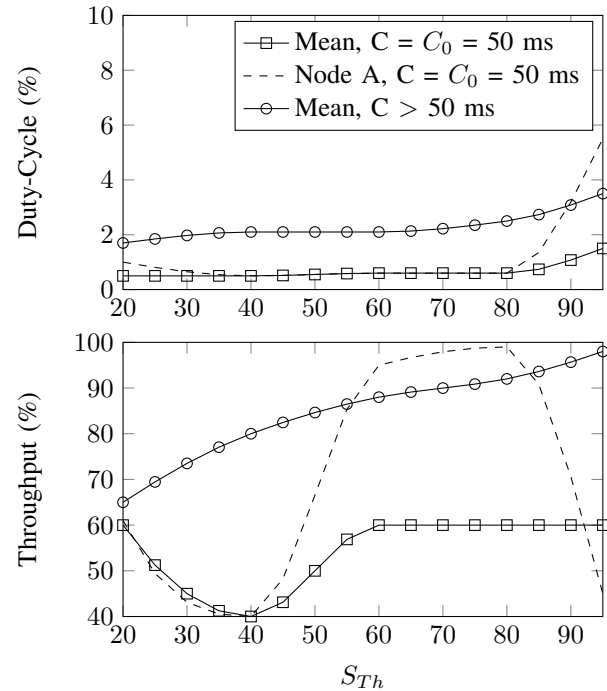
We can also see that the duty-cycle of the node is nearly 100% when $S_{Th} = 95$ as compared to other lower values of S_{Th} . This clearly shows that to achieve synchronicity with a high threshold the node must remain awake slightly longer as compared to the low threshold in the *synchronization state*. In addition, note that, in this set of experiments, we perform neighbourhood size calculations only in the first T period and we observe that due to lossy wireless links, this value might not be the actual neighbourhood size. For this reason, we selected the average number of neighbours by keeping nodes awake 100% for the first $5T$ in the following experiments.

C. Effects of the Broadcast Message Interval, T

Figure 10 presents the effects of T on the duty-cycle (%) and throughput (%). We use the three values of $T = 10$ s, 20 s, 30 s, and varied S_{Th} from 20 to 95. We found that the duty-cycle increases nearly 0.1% with every 10 increase in S_{Th} till $S_{Th} = 60$ for all values of T . However, for $S_{Th} > 60$, its incremental effect on the duty-cycle is slightly higher (1–3%)



(a)



(b)

Fig. 9: (a) The Motelab test-bed topology where nodes are shown as small circles with radio coverage shown by large circles; (b) we study the behavior of individual node A (radio coverage of node A shown by the shaded circle) in the given topology, where all nodes run EBS with different synchronization thresholds (S_{Th}); we also analyze the average duty-cycle and average throughput percentages of all nodes when $C = C_0 = 50$ ms and when C is adaptive according to Equation (10).

for low values of T , as compared to high values of T . On the other hand, throughput is not much affected by T ; higher values of T perform better for low S_{Th} and the lower values of T perform better for high S_{Th} . However, there is a gradual increase in throughput from 20–30% to 80–90% when S_{Th} varies from 20 to 95, respectively.

D. Comparative Performance

We compare our result with *Refractory Periods* [32] because it is the closest of the bio-inspired synchronization approaches to ours. To make it suitable for duty-cycling, we use the $T_{Ref} = T/2$, as suggested in paper [10] for achieving quick loose synchronicity, we call this the Modified Refractory Period (MRF).

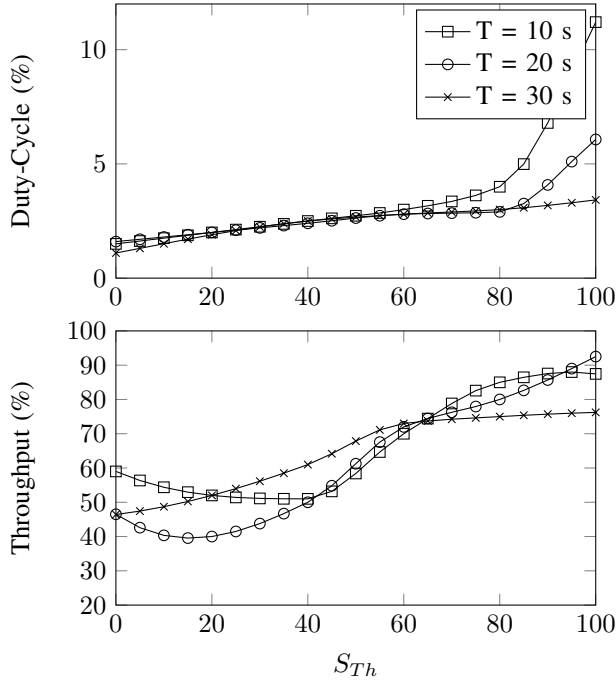


Fig. 10: EBS duty-cycle and throughput with adaptive C .

From Figure 10, it is clear that for our current network topology, setting $S_{Th} = 80$ achieves more than 85% throughput while keeping duty-cycle below 5%, which means saving nearly 80% energy by avoiding idle listening. We show that EBS achieves almost similar throughput, but with a large improvement in duty-cycling as shown in Figure 11. To compare our results with MRF, we choose $S_{Th} = 80$.

In Figure 11, we use adaptive C that is calculated by putting $C_0 = 50$ ms in Equation (10), however, if we use value of adaptive C calculated when $C_0 = 100$ ms, we can achieve the same or higher throughput compared to MRF. Therefore, in this experiment we present the throughput achieved at the minimum duty-cycle for adaptive C (calculated using $C_0 = 50$ ms), which EBS can achieve while accommodating propagation and processing delays that are caused by the lower layers (e.g., the CSMA MAC layer).

VIII. SUMMARY

In this paper, we presented EBS, a fully decentralized scheme for wireless embedded and IoT systems that ensures network-wide message broadcast in duty-cycled networks without requiring costly global synchronization. We have identified the challenges of providing such a mechanism, formalized the constraints therein and produced algorithms that are lightweight and robust to changing network topologies

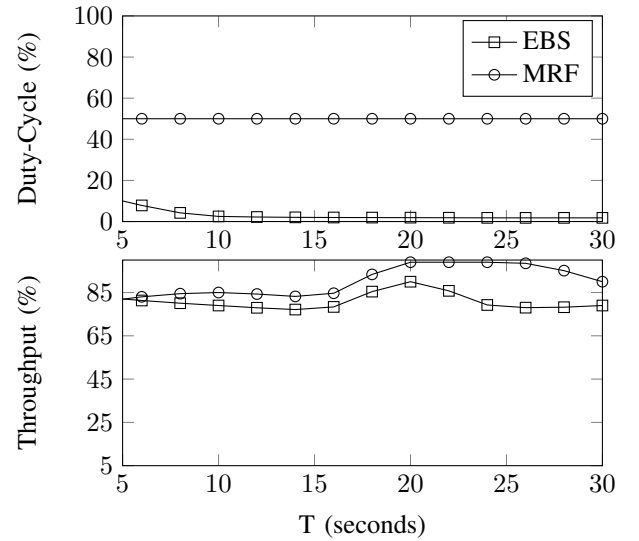


Fig. 11: Comparative duty-cycle and throughput of EBS (with adaptive C and $S_{Th} = 80$) and MRF.

and failures. EBS has been designed to provide support for broadcast messaging that uses local synchronization within a duty-cycled network. Further, EBS has been fully implemented on a test-bed to show the affects of convergence parameters and comparative performance with the nearest state-of-the-art protocol (MRF). Here, we have showed that EBS achieves more than 80% throughput while keeping the duty-cycle down to less than 5% (for broadcast message intervals, $T > 10$ s). Therefore, the results confirmed that EBS can match the throughput of comparative schemes that are continuously awake (100% duty-cycle) but with a considerable reduction in the network-wide energy consumption.

ACKNOWLEDGMENT

This work was supported by UK-India Education and Research Initiative (UKIERI) award and United Kingdom NERC research grant (NE/I00694X/1). Tiago Periera was partially supported by FAPESP CeMEAI grant 2013/07375-0, Russian Science Foundation grant 14-41-00044 at the Lobayevsky University of Nizhny Novgorod and by the European Research Council - AdG grant number 339523 RGDD.

REFERENCES

- [1] Btnode: Datasheet. [Accessed on 17th August 2008]. [Online]. Available: <http://www.btnode.ethz.ch/pub/uploads/Main/>
- [2] Micaz: Datasheet. [Accessed on 17th August 2008]. [Online]. Available: <http://www.xbow.com/Products/Product>
- [3] Sunspot: Datasheet. [Accessed on 17th August 2008]. [Online]. Available: <http://www.sunspotworld.com/docs/Green/SunSPOT-TheoryOfOperation.pdf>
- [4] P. Zhang, H. Yao, and Y. Liu, "virtual network embedding based on computing, network and storage resource constraints," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.
- [5] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On optimal scheduling in duty-cycled industrial iot applications using ieee802.15.4e tsch," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3655 – 3666, Oct. 2013, doi:10.1109/JSEN.2013.2266417.
- [6] A. Athreya and P. Tague, "Network self-organization in the internet of things," in *the Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2013, pp. 25–33.

- [7] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *the Second International Conference on Embedded Networked Sensor Systems (SenSys'04)*. ACM, 2004, pp. 39–49.
- [8] P. Yadav, N. Yadav, and S. Varma, "Cluster based hierarchical wireless sensor networks (chwsn) and time synchronization in chwsn," in *2007 International Symposium on Communications and Information Technologies*, Oct 2007, pp. 1149–1154.
- [9] L. Xu, R. Collier, and G. M. P. O'Hare, "A survey of clustering techniques in wsns and consideration of the challenges of applying such to 5g iot scenarios," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.
- [10] A. Tyrrell, G. Auer, and C. Bettstetter, "Emergent slot synchronization in wireless networks," *Transactions on Mobile Computing. IEEE.*, vol. 9, no. 5, pp. 719–732, 2010.
- [11] D. G. Reina, S. L. Toral, F. Barrero, N. Bessis, and E. Asimakopoulou, "The role of ad hoc networks in the internet of things: A case scenario for smart environments," *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence*, vol. 460, pp. 89 – 113, Jan. 2013.
- [12] Ocado: Online grocery warehouse robots. [Accessed on 23rd Feb 2016]. [Online]. Available: <http://www.v3.co.uk/v3-uk/feature/2447945/ocado-builds-an-robot-army-to-do-your-shopping>
- [13] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, and J. Lloret, "Context-aware cloud robotics for material handling in cognitive industrial internet of things," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.
- [14] E. Yanmaz, M. Quaritsch, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Communication and coordination for drone networks," *Ad Hoc Networks, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 184, pp. 79–91, Dec. 2016.
- [15] P. Yadav and J. A. McCann, "Ebs: Decentralised slot synchronisation for broadcast messaging for low-power wireless embedded systems," in *the 5th International Conference on Communication System Software and Middleware (COMSWARE'11)*. ACM, July 2011, pp. 9:1–9:6.
- [16] K.-Y. Cheng, K.-S. Lui, Y.-C. Wu, and V. Tam, "A distributed multi-hop time synchronization protocol for wireless sensor networks using pairwise broadcast synchronization," *Transactions on Wireless Communications. IEEE.*, vol. 8, no. 4, pp. 1764–1772, April 2009.
- [17] B. Geller, "Advanced synchronization techniques for the internet of things," in *International Symposium on Signal, Image, Video and Communications (ISIVC'16)*, Nov 2016, pp. 180–184.
- [18] W. Su and I. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *Transactions of Networking. IEEE/ACM.*, vol. 13, no. 2, pp. 384–397, April 2005.
- [19] J. Albrecht, C. Tuttle, A. C. Snoeren, and A. Vahdat, "Loose synchronization for large-scale networked systems," in *Proceedings of the Annual Technical Conference on USENIX'06*, ser. ATEC'06. Berkeley, CA, USA: USENIX Association, 2006, pp. 28–28. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1267359.1267387>
- [20] R. Zhang and J. T. Kwok, "Asynchronous distributed admm for consensus optimization," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–1701–II–1709. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3044805.3045082>
- [21] H. Yan, Y. Zhang, Z. Pang, and L. D. Xu, "Superframe planning and access latency of slotted mac for industrial wsn in iot environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1242–1251, May 2014.
- [22] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *the International Conference on Information Processing in Sensor Networks (IPSN'09)*. ACM, 2009, pp. 37–48.
- [23] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *the 3rd international conference on Embedded networked sensor systems*. ACM, 2005, pp. 142–153.
- [24] I. Bojic, T. Lipic, and M. Kusek, "Scalability issues of firefly-based self-synchronization in collective adaptive systems," in *IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, Sept 2014, pp. 68–73.
- [25] I. Bojic and K. Nymoen, "Survey on synchronization mechanisms in machine-to-machine systems," *Eng. Appl. Artif. Intell.*, vol. 45, no. C, pp. 361–375, Oct. 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.engappai.2015.07.007>
- [26] M. Petrocchi, A. Spognardi, and P. Santi, "Bioinspired security analysis of wireless protocols," *Mob. Netw. Appl.*, vol. 21, no. 1, pp. 139–148, Feb. 2016. [Online]. Available: <http://dx.doi.org/10.1007/s11036-016-0702-z>
- [27] A. Tyrrell, G. Auer, C. Bettstetter, and R. Naripella, "How does a faulty node disturb decentralized slot synchronization over wireless networks?" in *the International Conference on Communications (ICC'10)*. IEEE, 2010, pp. 1–5.
- [28] R. E. Mirollo and S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators," *Journal of Applied Mathematics. SIAM.*, vol. 50, no. 6, pp. 1645–1662, 1990.
- [29] P. Closas, E. Calvo, J. Fernandez-Rubio, and A. Pages-Zamora, "Coupling noise effect in self-synchronizing wireless sensor networks," in *the Eighth Workshop on Signal Processing Advances in Wireless Communications (SPAWC'07)*. IEEE, 2007, pp. 1–5.
- [30] J. Klinglmayr, C. Bettstetter, and M. Timme, "Globally stable synchronization by inhibitory pulse coupling," in *the Second International Symposium on Applied Sciences in Biomedical and Communication Technologies*. ACM, 2009, pp. 1–4.
- [31] O. Babaoglu, T. Binci, M. Jelasity, and A. Montresor, "Firefly-inspired heartbeat synchronization in overlay networks," in *the First International Conference on Self-Adaptive and Self-Organizing (SASO'07)*. ACM, July 2007, pp. 77–86.
- [32] J. Degesys, P. Basu, and J. Redi, "Synchronization of strongly pulse-coupled oscillators with refractory periods and random medium access," in *the Symposium on Applied computing (SAC'08)*. ACM, 2008, pp. 1976–1980.
- [33] Y. Wang, F. Nunez, and F. Doyle, "Statistical analysis of the pulse-coupled synchronization strategy for wireless sensor networks," *Transactions on Signal Processing. IEEE.*, vol. 61, no. 21, pp. 5193–5204, 2013.
- [34] W. Masood, J. Klinglmayr, and C. Bettstetter, "Experimental evaluation of pulse-coupled oscillator synchronization in ieee 802.15.4 networks," in *the Fourth International Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications.*, ser. DIVANet'14. New York, NY, USA: ACM, 2014, pp. 145–151. [Online]. Available: <http://doi.acm.org/10.1145/2656346.2656360>
- [35] R. Leidenfrost and W. Elmenreich, "Firefly clock synchronization in an 802.15.4 wireless network," *Journal of Embedded Systems. EURASIP.*, vol. 2009, pp. 7:1–7:17, Jan 2009. [Online]. Available: <http://dx.doi.org/10.1155/2009/186406>
- [36] L. Cui and H. Wang, "Reachback firefly synchronicity with late sensitivity window in wireless sensor networks," in *the Ninth International Conference on Hybrid Intelligent Systems (HIS'09)*. IEEE Computer Society, 2009, pp. 451–456.
- [37] R. Pagliari and A. Scaglione, "Scalable network synchronization with pulse-coupled oscillators," *Transactions on Mobile Computing. IEEE.*, vol. 10, no. 3, pp. 392–405, 2011.
- [38] P. Yadav and J. A. McCann, "Ya-mac: Handling unified unicast and broadcast traffic in multi-hop wireless sensor networks," in *the International Conference on Distributed Computing in Sensor Systems (DCOSS'11)*, no. 9. IEEE, 2011, pp. 1–9.
- [39] M. C. Guenther, P. Yadav, J. T. Bradley, and J. A. McCann, "Model based optimization of ebs-mac," in *Sigmatrics Performance*. ACM, 2012.
- [40] P. Yadav and J. A. McCann, "Qos based event delivery for disaster monitoring applications," in *the Fifth IEEE Conference on Wireless Communication and Sensor Networks (WCSN)*. IEEE, Dec 2009, pp. 2–9.
- [41] E. Gelenbe, E. Ngai, and P. Yadav, "Routing of high priority packets in wireless sensor networks," in *SPIE Defense, Security, and Sensing (SPIE DSS'09)*. SPIE, April 2009, pp. 1–6.
- [42] P. Yadav and J. A. McCann, "Qos and congestion control for pervasive event driven applications," in *ACM International Conference on Pervasive Services (ICPS)*. ACM, July 2010, pp. 1–6.
- [43] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *the 23rd Annual International Conference on Computer Communications (INFOCOM'02)*, vol. 3. IEEE, 2002, pp. 1567–1576.
- [44] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle mac with scheduled channel polling," in *the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06)*. ACM, 2006, pp. 321–334.
- [45] T. Pereira, J. Eldering, M. Rasmussen, and A. Veneziani, "Towards a general theory for coupling functions allowing persistent synchronisation," *Nonlinearity*, vol. 27, no. 3, pp. 501–525, Feb. 2014.
- [46] T. Pereira, D. Eroglu, G. B. Bagci, U. Tirnakli, and H. J. Jensen, "Connectivity-driven coherence in complex networks," *Physical Reviews Letters*, vol. 110, no. 23, p. 5, June 2013.

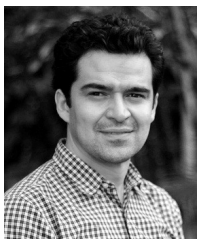
- [47] K. Klues, G. Hackmann, O. Chipara, and C. Lu, "A component-based architecture for power-efficient media access control in wireless sensor networks," in *the Fifth International Conference on Embedded Networked Sensor Systems (SenSys'07)*. ACM, 2007, pp. 59–72.
- [48] Motelab: Sensor network testbed. [Accessed on 17th August 2008]. [Online]. Available: <http://www.btnode.ethz.ch/pub/uploads/Main/>



Poonam Yadav is a research and teaching associate at the Computer Laboratory, University of Cambridge. She received the PhD degree in Computing from Imperial College London, in 2011 and M.Tech from IIT, Allahabad, India, in 2007. She is a recipient of UK-India Education and Research Initiative (UKIERI) PhD Award and has worked on various NERC, TSB, EU, EPSRC, IBM and Microsoft funded research projects and author of over 30 papers in Distributed Systems, Social Computing, Sensor Systems and IoT. She is currently the Chair of ACM-W UK professional Chapter and a member of ACM, IEEE and BCS.



Julie A. McCann is a professor in computer systems with Imperial College. Her research centers on highly decentralized and self-organizing scalable algorithms for spatial computing systems, e.g., wireless sensing networks. She leads both the Adaptive Embedded Systems Engineering Research Group and the Intel Collaborative Research Institute for Sustainable Cities and is currently working with NEC and others on substantive smart city projects. She has received significant funding from bodies such as the United Kingdom's EPSRC, TSB, and NERC as well as various international funds, and is an elected peer for the EPSRC. She has actively served on, and chaired, many conference committees and is currently associative editor of the ACM Transactions on Autonomous and Adaptive Systems. She is a fellow of the BCS.



Tiago Pereira is a professor of Mathematics at Institute of Mathematics and Computer Science, University of Sao Paulo, Brazil and author of over 40 papers on complex networks and dynamical systems. He is a principal investigator in the Center for Industrial Mathematics funded by FAPESP in partnership with leading companies. The main thrust of his research is the collective dynamics of complex systems concerning both theory and data-driven approaches. He obtained his PhD from Potsdam University in Germany, and held postdoctoral positions in Berlin, Sao Paulo and was a Leverhulme and Marie Curie Fellow at Imperial College London.