C-Sync: Counter-based Synchronization for Duty-cycled Wireless Sensor Networks

C-Sync: Counter-based Synchronization for Duty-cycled Wireless Sensor Networks

Kok-Poh Ng, Charalampos Tsimenidis, Wai Lok Woo

School of Electrical and Electronic Engineering, Newcastle University

Abstract

Different variants of synchronous duty-cycle MAC protocols have been designed for wireless sensor networks to reduce energy consumption. However, the synchronization process of these protocols remains a significant contributor to the energy consumption. In this paper, a new energy-efficient synchronization algorithm referred to as C-Sync is proposed. C-Sync reduces energy consumption by adaptively regulating the synchronization traffic and synchronization wakeup period based on the changing network neighborhood conditions through counter-based and exponential-smoothing algorithms. Extensive simulations of multi-hop multi-neighborhood network scenarios are performed using ns-2. We compare C-Sync with the fixed periodic synchronization (F-Sync) algorithm and the 1-Sync algorithm and show that C-Sync outperforms F-Sync and 1-Sync in energy-efficiency over a wide range of node densities, drift rates and duty cycles.

Keywords: duty-cycle, MAC protocols, energy-efficient, synchronization, counter-based

1. Introduction

A typical wireless sensor network (WSN) generates very light traffic and sensor nodes spend most of the time listening to the radio channel and idling. This idle listening is the dominant source of energy consumption in WSNs.

Duty-cycling is a mechanism where sensor nodes alternate between active and sleep periods. This is a common approach used in the MAC layer to reduce idling time and consequently reduce energy consumption. Sensor nodes in duty-cycle networks schedule the transmission and reception of data during active periods, and switch the radios off completely during sleep periods to conserve energy. References Huang et al. [1] and Ahmad et al. [2] provide comprehensive reviews of duty-cycle MAC protocols for WSNs. In addition, energy-efficient solutions in Network, Transport and Application layers are also discussed in Akyildiz and Vuran [3].

There are two main categories of duty-cycle MAC protocols. The synchronous approach makes use of a MAC layer synchronization algorithm to synchronize sensor nodes in the same neighborhood so that they can wake up at the same time to exchange sensor data. On the other hand, the asynchronous or preamble sampling approach does not use a synchronization algorithm, but places the burden of data delivery on the senders. When a sensor node has data to send, it has to first transmit a preamble that is longer than the sleep period of the receiver so that the receiver will be able to detect the preamble. Once the preamble is detected, the receiver will stay awake to receive the data.

The seminal synchronous duty-cycle MAC protocol S-MAC Ye et al. [4] further divides the active periods into SYNC and DATA windows. During SYNC windows, sensor nodes broadcast synchronization (*sync*) packets periodically to synchronize the clocks of the neighboring nodes. During DATA windows, sensor nodes send out data packets from the higher layers based on some contention mechanisms to avoid collisions. Later developments of synchronous MAC protocols such as DW-MAC Sun et al. [5], AS-MAC Zhao et al. [6], SEA-MAC Zhao et al. [7] and LO-MAC Nguyen et al. [8] focus on improving the energy efficiency, throughput and delay performances by implementing changes in the scheduling and transmission of data packets, leaving the synchronization algorithm largely unchanged.

The synchronization algorithm adopted by the above synchronous MAC protocols is based on a fixed, periodic synchronization packet broadcast algorithm (F-Sync) Ye et al. [9] by the sensor nodes in SYNC windows. This algorithm works fine when the network is sparse. When the network is dense, however, too many sync packets are generated and transmitted in the network, causing collisions and increasing energy consumption unnecessarily. The energy consumption for the synchronization process in SYNC windows is not insignificant. For many of the synchronous MAC protocols implemented, the duration of the SYNC window is about one-third of the entire active period. It is therefore worthwhile to examine the synchronisation process of duty-cycle MAC in more detail.

In this paper, we present a new, counter-based synchronization algorithm (C-Sync) that works in the framework of synchronous MAC protocols. C-

Sync reduces energy consumption and improves the effectiveness of the synchronization process by adaptively switching off the radio, as well as reducing unnecessary *sync* packet transmission when the network density is high. It also enables the sensor nodes to wake up more frequently to receive *sync* packets when the network density is low. Extensive simulations are conducted for multi-hop multi-neighborhood grid networks with different densities for performance evaluation. Effects of drift and duty-cycling on both the energy and data performance are also studied.

The contributions of this work can be summarized as follows:

- We introduced and developed a new synchronization algorithm that is adaptive to a wide range of network densities. This algorithm effectively reduces congestions and collisions in the network when *sync* traffic is high and maintains synchronization performance when *sync* traffic is low. In a large multi-hop WSN, different neighborhoods have different densities; an adaptive synchronization algorithm will enable the MAC protocol to deliver a better energy performance.

- End-to-end multi-hop network simulations are performed against sensor nodes with different duty cycles, network densities and clock drifts to analyze the sensitivity and stability of the synchronization algorithms. Both energy and data performance are evaluated and analyzed.

- Although we implemented the C-Sync algorithm with S-MAC, it is also compatible with other synchronous duty-cycle MAC protocols such as DW-MAC, AS-MAC, etc.

The rest of this paper is organized as follows: Section 2 discusses the related work of the synchronization algorithms proposed for duty-cycle WSN. Section 3 discusses the synchronization process in a duty-cycle WSN and the effects of different network conditions. Section 4 describes the design of the proposed C-Sync algorithm, detailing the sync transmission and reception processes. In section 5, the performances of C-Sync are evaluated based on the simulation results, including a comparison with F-Sync and 1-Sync. Finally, section 6 concludes the paper.

2. Related Work

Time synchronization in WSNs can be achieved by exchanging timing messages among the sensor nodes. There are, broadly categorized, three approaches for time synchronization in WSNs. They are the sender-receiver synchronization, the one-way message dissemination (or unidirectional reference broadcast) and the receiver-receiver synchronization Wu et al. [10]. In the unidirectional reference broadcast approach Maróti et al. [11] Shannon et al. [12], a single-message broadcast carrying reference clock signal is used to achieve local synchronization with the participating nodes in the sender neighborhood. On the other hand, both sender-receiver Ganeriwal et al. [13] Ahmad et al. [14] and receiver-receiver Elson et al. [15] Gong and Sichitiu [16] synchronizations use multiple message exchanges to achieve pair-wise synchronization with high accuracy. A comprehensive comparison and review of different synchronization algorithms in WSNs can be found in Youn [17].

The listen period in synchronous duty-cycle MAC protocols is much longer than the clock drift. As such, a much looser synchronisation among neighboring nodes is required compared with TDMA schemes with very short timeslots Ye et al. [9]. In addition, as the frame structure of synchronous duty-cycle MAC provides only small time windows for exchanging timing messages, single-message unidirectional broadcast is the most appropriate and energy efficient among the three approaches for synchronizing sleep/wakeup schedules of the sensor nodes.

F-Sync was proposed in Ye et al. [9] together with S-MAC protocol and has since been the default synchronization algorithm used in the synchronous MAC protocols that were developed later. As the neighboring nodes need to coordinate their sleep/wakeup schedules, and the clock for each sensor node drifts independently from one another, the drifts can cause data loss if the clocks are left unsynchronized. Using the F-Sync algorithm, each sensor node broadcasts a *sync* packet periodically to update the neighbors on its sleep/wakeup schedule to prevent long-term phase offset.

Intelligent Network Synchronization (INS) Sthapit and Pyun [18] attempts to improve the energy efficiency in the synchronization process by exploiting the periodic nature of *sync* packet transmission in F-Sync. In the INS networks, each node maintains a counter for each of its neighbors. Each counter is increased by one after every cycle. When a node receives a *sync* packet from a neighbor, the corresponding counter will be reset to zero. By examining the list of its counters, the node is able to determine whether there will be a *sync* packet arriving in the current SYNC window. If any of the counter value is greater than or equal to the synchronization period, the node wakes up in the current window as it is expecting a *sync* packet to arrive. It will otherwise go to sleep to conserve energy. INS was simulated and evaluated over a linear and a sparse grid network with good energy performance. However, in a dense network where collisions frequently occur, the periodicity of *sync* packets from each neighbor cannot be guaranteed and INS faces the similar energy inefficiencies as F-Sync Ng et al. [19].

The 1-Sync algorithm proposed in Ng et al. [19] puts a sensor node to sleep after it receives a valid *sync* packet in the current synchronization period, which is independent of network neighborhood density. Similar to both F-Sync and INS, a sensor node using 1-Sync schedules a *sync* packet periodically. After a *sync* packet is sent, the sensor node stays awake during the subsequent SYNC windows and waits for a valid *sync* packet from its neighbors. Once a valid packet is received, the node goes into a synchronized state and will go to sleep in the subsequent SYNC windows to conserve energy. The sensor node will only turn its radio on when it is ready to transmit its *sync* packet, and the cycle repeats. 1-Sync is shown to be more energy efficient than F-Sync and INS in single neighborhood networks of wide density ranges. However, the performances of these synchronization algorithms under multi-hop multi-neighborhood networks in the presence of hidden terminals are not studied.

3. Synchronization in Duty-Cycle MAC Networks

3.1. Overview

The basic operation of common listen/sleep periods in synchronous duty-cycle MAC protocols requires a network-wide synchronization mechanism to synchronize the local clocks of the sensor nodes within the same neighborhood. Beacons or *sync* packets must be received regularly within a specific interval before the local clock drifts out of synchronization.

The design goal of the proposed synchronization algorithm design is to offer energy consumption efficiency while providing good data performance, including packet delivery ratio and end-to-end packet delay across a wide range of multi-hop WSNs. In this paper, we will study the performances of the proposed algorithm against F-Sync and 1-Sync algorithms in WSNs with different network densities, clock drifts and duty cycles. We will also compare the robustness and stability of these three synchronisation algorithms by examining individual node energy performance under a wide range of network scenarios.

Since the original S-MAC and many of the synchronous duty-cycle MAC family have implemented F-Sync for synchronization, it is important to have a brief description of F-Sync before we described the proposed C-Sync algorithm.

3.2. The operation of F-Sync

The operation cycle of a duty-cycle MAC protocol is divided into listen periods and sleep periods as shown in Figure 1.

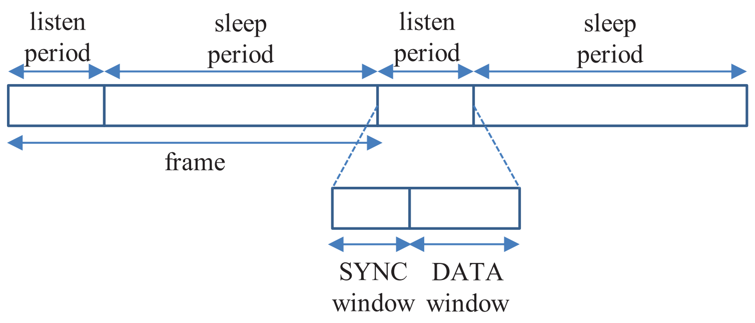

Figure 1: Synchronous duty-cycle MAC frame structure

The listen period is further divided into two distinct intervals, the SYNC window and the DATA window. Synchronization process takes place only in the SYNC windows. Each sensor node in an F-Sync network schedules a *sync* packet periodically to broadcast to the neighbors its sleep/wakeup schedule. At the beginning of each SYNC window, a sensor node wakes up to either transmit or receive a *sync* packet. If a synchronization period of N_{SP} frames has elapsed since the last time the node sent a *sync* packet, it will schedule a new *sync* packet and follow a contention procedure for the transmission of the packet. During the contention window, if the medium is busy, it will revert to packet receiving mode and postpone the *sync* transmission to the next SYNC window. Upon receiving a *sync* packet, the node re-synchronizes its sleep/wakeup schedule with the time information given in the *sync* packet. It stays idle for the entire SYNC window if there is no packet to transmit or receive. At no time will a sensor node go to sleep during SYNC windows. Figure 2 illustrates the F-Sync operation in an N-node neighborhood ($N < N_{SP}$).

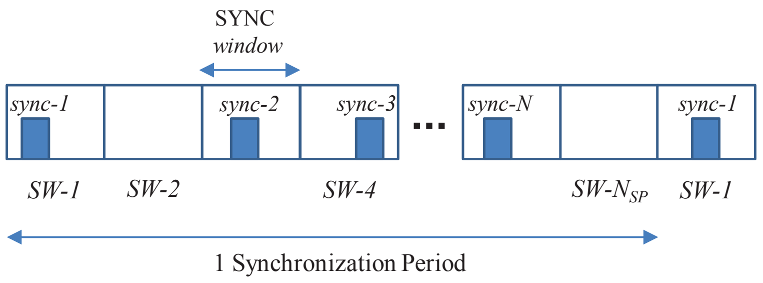

Figure 2: Illustration of *sync* packet transmissions in SYNC windows for the N-node neighborhood in an F-Sync network ($N < N_{SP}$)

3.3. Network Density and Sync Collision

Consider an N-node neighborhood in the absence of hidden or exposed terminals. When the density is low ($N < N_{SP}$), there are more SYNC windows available than the number of nodes in the neighborhood. There could be some collisions in the initial periods when the nodes schedule their *sync* transmission in the same window. However, in the steady state, each node eventually settles into its own unique windows for periodic *sync* transmissions with no collision.

In a high density neighborhood ($N > N_{SP}$), the number of *sync* packets scheduled to be transmitted within the synchronization period is more than the number of SYNC windows available. Consequently, one of the following three scenarios may occur when a sensor node is trying to broadcast a *sync* packet:

1. Only one *sync* packet is scheduled in the current window and it is transmitted successfully.
2. Two or more nodes transmit their *sync* packets in the same timeslot in current window, resulting in a *sync* collision.
3. The *sync* packets are scheduled in different timeslots, and those scheduled in later timeslots will postpone their *sync* transmissions to the next SYNC window upon detection of the first *sync* transmission.

As density increases, *sync* traffic increases, which causes the increase in the probabilities of *sync* collision (scenario 2), and *sync* postponement (scenario 3). Both *sync* collisions and *sync* postponements are undesirable as they consume energy unproductively. Analysis and simulations of *sync* collision probabilities at different densities were performed in Ng et al. [19]. To reduce the energy consumption, both 1-Sync and INS modify the *sync* reception process but make no change to the *sync* transmission process. As a result, they face the same *sync* congestion problem as F-Sync in high density neighborhoods.

In addition, in a multi-hop sensor network, there are multiple synchronization neighborhoods with the following additional complexities:

1. Sensor nodes in different regions of the network are in neighborhoods of different densities. Inter-arrival times between *sync* packets received by each node can have large variations.
2. *Sync* collision is unavoidable even at unsaturated neighborhoods due to the presence of hidden terminals.
3. Transmit and receive *sync* intervals are not periodic due to the interactions among the different neighborhoods.

It is therefore desirable to have an adaptive synchronisation algorithm that enables sensor nodes to dynamically adjust their *sync* transmissions in the different neighborhoods to maintain a desirable level of synchronization performance.

3.4. Clock Drift Rate

Every sensor node has a local clock that works independently. Due to the imperfections of clock oscillators, the time maintained by each sensor node will drift away from the ideal time, as well as from one another, over time. As discussed in Wu et al. [10], the clock function for a node i can be generally modelled as

$$C_i(t) = \theta + f_i \cdot t \tag{1}$$

where θ and f_i are the clock offset (phase difference) and clock drift rate (frequency difference) respectively. Therefore if the two nodes A and B are initially synchronized with each other, then the time difference between the two clocks in time t is shown as

$$C_A(t) - C_B(t) = (f_A - f_B) \cdot t \tag{2}$$

which is proportional to the time elapsed since the last synchronization. Periodic re-synchronization is thus required to prevent the continuing increase of clock offsets that will affect communication reliability and energy consumption efficiency.

A typical crystal-quartz oscillator commonly used in sensor networks has a drift rate of up to 40 parts per million (±40 ppm). In addition, external factors such as temperature, voltage changes and hardware aging also add to the clock drift. Therefore a duty-cycle MAC protocol and its synchronization algorithm must be able to handle different levels of clock drifts and still provide good energy and data performance.

3.5. Duty Cycle

Duty cycling is one of the key mechanisms in WSN to reduce energy wastage in idle listening and improve network lifetime. In general, lower duty-cycle networks have lower energy consumptions albeit at the expense of longer packet delivery times. With longer sleep periods and longer frame times in low duty-cycle operations, *sync* packet inter-arrival times are also longer and clock synchronization becomes a greater challenge. For example, a 1Mbps S-MAC frame is about 8.0 seconds and 1.6 seconds in 2% and 10% duty-cycle (dc) networks respectively. With 10-frame synchronization period, a node in a 2% dc network will schedule a *sync* packet every 80 seconds compared to just 16 seconds in a 10% dc network. Therefore in comparing different synchronization algorithms, it is important to evaluate the stability of their performance in different duty-cycle operations.

3.6. Sync Scheduling and Broadcast Methods

The overloading of *sync* traffic in high density neighborhoods is similar to the well-known broadcast storm problem Tseng et al. [20] although the former is due to the generation and transmission of new single-hop broadcast packets and the latter is due to the retransmission of the same multi-hop broadcast packets.

Different broadcasting methods to reduce broadcast storm in ad hoc wireless networks reviewed in Koscielnik and Stepien [21] broadly classifies them into probabilistic-based, area-based, neighbor knowledge and multipoint relay methods. Area-based methods require the nodes to have estimations of distances to or locations of the neighbors; neighbor knowledge methods require the nodes to collect information about the neighbors via periodic Hello packets; and multipoint relay methods are specific to multi-hop broadcasts.

There are two approaches in the probability-based methods: the use of a probability value p and the use of counter-value c for re-transmission decisions. The counter-based approach Tseng et al. [22], when adapted to our *sync* transmission scheduling, has the desirable characteristics of automatically regulating *sync* traffic with different neighborhood densities.

4. C-Sync Algorithm Design

The proposed C-Sync algorithm operates in the SYNC windows of a synchronous duty-cycle MAC protocol. There are two sub-algorithms in C-Sync, a counter-based *sync* transmission algorithm that reduces *sync* load and energy consumption when the neighborhood is dense, and an adaptive exponential-smoothing *sync* reception algorithm that improves *sync* performance when the network is sparse.

4.1. Counter-based Algorithm for Sync Transmission

Similar to F-Sync, and 1-Sync, a C-Sync node schedules the next *sync* packet N_{SP} frames after it has successfully transmitted the current *sync* packet. However, when the *sync* transmission is unsuccessful, the F-Sync and 1-Sync algorithms will attempt to transmit the scheduled *sync* packets in every subsequent SYNC window until they are successfully transmitted. In a high density neighborhood where the number of *sync* packets scheduled is more than the number of SYNC windows available in a synchronization period, many *sync* packets scheduled will be withheld, and the congestion remains a problem. C-Sync, on the otherhand, provides a mechanism to cancel the scheduled *sync* packets when transmission is unsuccessful, reducing the traffic load when the neighborhood gets congested.

When a *sync* packet is first scheduled, a C-Sync node initiates a counter c_{sn} to count the number of valid sync packets received while waiting for its turn to transmit. The sensor node attempts to transmit its scheduled *sync* packet following a contention procedure. If it is unsuccessful, the counter c_{sn} is incremented by 1. If the value of c_{sn} is less than the counter threshold C_{thres}, the *sync* packet is postponed and rescheduled to the next SYNC window. Otherwise, the scheduled *sync* packet is cancelled, and the sensor node then goes to sleep and a new *sync* packet will be scheduled N_{SP} frames (one synchronization period) later. The counter-based *sync* transmission scheme implemented in C-Sync is illustrated in Figure 3.

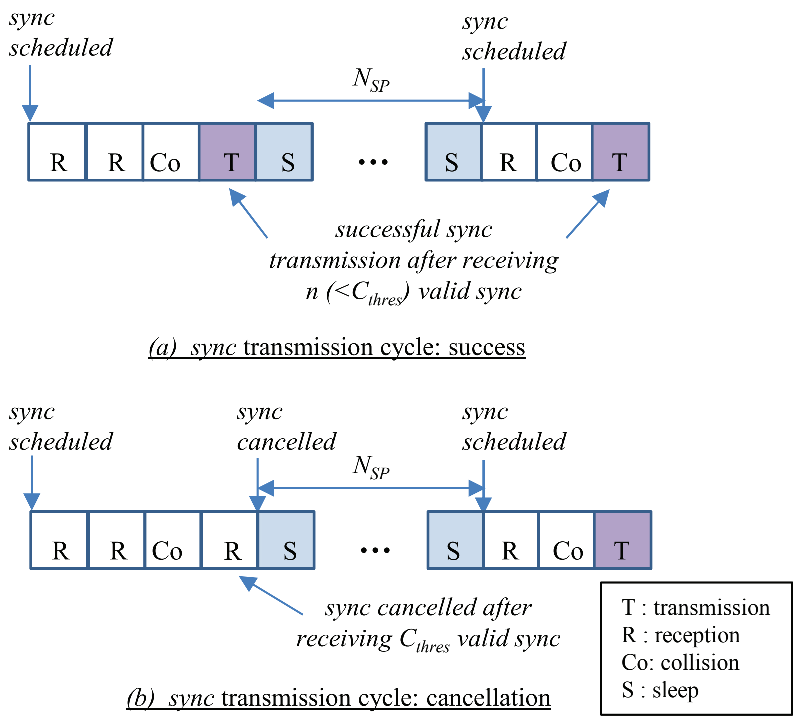
sync scheduled
N_{SP}
sync scheduled
R
R
Co
T
S
•••
S
R
Co
T
successful sync transmission after receiving n (<C_{thres}) valid sync
(a) sync transmission cycle: success
sync scheduled
sync cancelled
N_{SP}
sync scheduled
R
R
Co
R
S
•••
S
R
Co
T
sync cancelled after receiving C_{thres} valid sync
T : transmission
R : reception
Co: collision
S : sleep
(b) sync transmission cycle: cancellation

Figure 3: Illustration of C-Sync counter-based *sync* transmission with C_{thres}=3

When neighborhood density is low, *sync* traffic is low and there is a high probability of successful *sync* transmission without the need to trigger the *sync* packet cancellation algorithm. This is advantageous as each and every *sync* packet is important for neighborhood synchronization, since there are so few of them present in the neighborhood.

When neighborhood density is high, there are many more *sync* packets in the neighborhood. The probability of receiving C_{thres} *sync* packets increases which will trigger the algorithm to cancel the scheduled *sync* packets, and allows the node to go to sleep in the subsequent SYNC windows. This process reduces the *sync* load in the neighborhood, shortens the active waiting periods for *sync* transmission, and lowers the energy consumption of the sensor nodes.

4.2. Exponential-smoothing Algorithm for Sync Reception

A key mechanism of the energy conservation algorithm for synchronization is the ability to go to sleep in the SYNC windows as long as clock drift is within the tolerance limit of the synchronization. As the *sync* traffic load varies with neighborhood density, sensor nodes that wake up to receive a *sync* packet will have to wait for different intervals before they receive a valid *sync* packet. The waiting interval tends to be longer when the density is low and this will affect the synchronization performance.

The proposed C-Sync algorithm enables a sensor node to adjust its wake up interval dynamically to compensate for the waiting time it takes to receive a valid *sync* packet so that it has a higher chance of receiving a valid *sync* packet within a desired number of frames N_{RP} under different density and traffic conditions. A C-Sync node maintains a counter, w_{wk}, the number of SYNC windows it should be sleeping before waking up to receive *sync* packets. Upon waking up, it stays active in the SYNC windows and listens for *sync* packets. It will go to sleep after a valid *sync* packet is received. The waiting period for the arrival of a valid *sync* packet is denoted as w_a (frames). To maintain the received synchronization interval close to N_{RP}, w_{wk} should be compensated with w_a as follows:

$$w_{wk} = N_{RP} - w_a \tag{3}$$

The waiting interval w_a varies over time and is dependent on both the node density and the collision level in the neighborhood. Therefore it is necessary to forecast the next waiting time for the computation of the next wake-up interval. As the waiting time is not expected to show any trend, simple exponential-smoothing technique is appropriate and is used in the adaptive C-Sync *sync* reception algorithm. Using the exponential-smoothing technique is also advantageous as it is both memory- and computational-efficient. The computation for the wake-up interval at the $(k+1)^{th}$ period, $w_{wk}(k+1)$ can be formulated as

$$w_{wk}(k+1) = \lfloor \alpha\{N_{RP} - w_a(k)\} + (1-\alpha)w_{wk}(k) \rfloor, \tag{4}$$

where α is the smoothing factor which can be chosen between zero and one, and the initial value of the wake-up interval, $w_{wk}(0)$, is chosen to be the midpoint value of the desired received interval as

$$w_{wk}(0) = \lfloor \frac{N_{RP}}{2} \rfloor \tag{5}$$

The exponential-smoothing *sync* reception scheme implemented in C-Sync is illustrated in Figure 4.

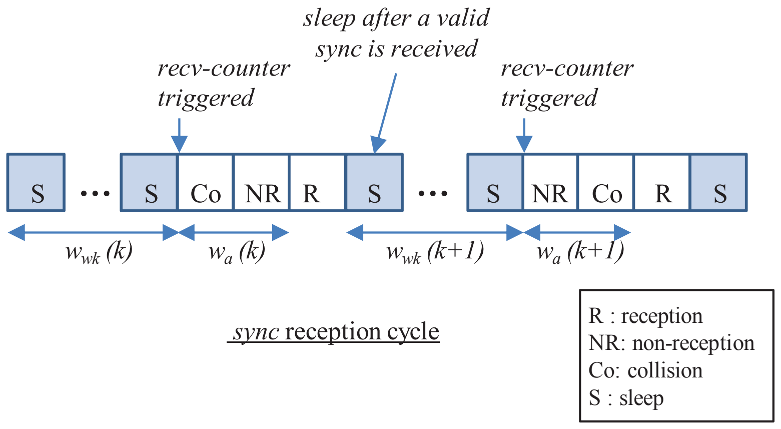

Figure 4: Illustration of C-Sync adaptive *sync* reception cycle

Combining the counter-based *sync* transmission and exponential-smoothing *sync* reception algorithms, C-Sync algorithm will be able to modify the sensor nodes' behaviour adaptively in a wide range of network neighborhood densities. The pseudo-code of the proposed C-Sync algorithm is shown in Figure 5

C-Sync Algorithm

```
initialization:
    syncTxCounter = $N_{SP}$
    nextRxWkUp = int ($N_{RP}$ / 2)
    rxWkUpCounter = nextRxWkUp
    syncWaitCounter = 0
    syncRecvCounter = 0

sync window begins:
    if (syncTxCounter == 0 or rxWkUpCounter == 0) {          // time to wake up
        wakeup()
        if (syncTxCounter == 0) {
            send_sync()                              // procedure to send sync packet
            if (send_success) {
                syncTxCounter = $N_{SP}$
                syncRecvCounter = 0
            }
        }
        if (rxWkUpCounter == 0) {
            if (sync_ received) {
                synchronise_node()                   // procedure to synchronize clock
                nextRxWkUp = int ($\alpha$ * ($N_{RP}$ – syncWaitCounter)
                                  + (1 - $\alpha$) * nextRxWkUp)
                rxWkUpCounter = nextRxWkUp
                syncWaitCounter = 0
                if (syncTxCounter == 0) {
                    syncRecvCounter++                // count number of sync packets received
                    if (syncRecvCounter == $C_{thres}$) {
                        syncTxCounter = $N_{SP}$          // cancel scheduled sync packet
                        syncRecvCounter = 0
                    }
                }
            else syncWaitCounter ++                  // increment sync waiting period counter
        }
    }
    else {
        if (syncTxCounter != 0) syncTxCounter --     // transmit counter countdown
        if (rxWkUpCounter != 0) rxWkUpCounter --     //receive wakeup counter countdown
sync window ends:
```

Figure 5: Adaptive C-sync algorithm with counter-based *sync* transmission and exponential- smoothing *sync* reception sub-algorithms

5. Performance Evaluation

In this section, we evaluate and compare the synchronization, energy and data performance of C-Sync against F-Sync and 1-Sync using ns-2 version 2.35. ns-2 is an object-oriented discrete event simulator developed through the VINT project by University of Southern California, and is by far the most popular simulator used in ad-hoc and sensor network simulations due to its flexibility and modularity Khan and Atif [23].

5.1. Simulation Setup

Multi-hop grid networks within an area of 500m x 500m are set up for the simulation of different synchronization scenarios. S-MAC, which is integrated in ns-2, is selected as the representative protocol for the duty-cycle MAC protocols in our simulations.

To represent networks of different densities, we started with a low density network of 9 nodes, forming 3x3 equal size square grids, and steadily increased to a high density of 49-node network forming 7x7 square grids, all in the same 500mx500m simulation area. Networks with different drift scenarios are also simulated. For each drift scenario, each sensor node has an independent local clock with a drift rate that is uniformly distributed between $\pm\triangle f$, where $\triangle f$ ranges from 0 ppm to 80 ppm. Simulations are also performed at four different duty cycles of 20%, 10%, 5% and 2%.

For each scenario, 30 independent simulation runs are performed over a period of 9000s. Constant bit rate (CBR) data traffic at 1 packet per minute is sent from the source node at one corner of the grid to the destination node at the diagonally opposite corner of the grid. The source node starts the data traffic at 100s after the start of the simulations to allow MAC layer protocol to stabilize, and stops the data traffic 60s before simulation ends. The size of the data packet and S-MAC protocol data unit (PDU) used are 100-byte and 120-byte respectively, so that each data packet can fit into 1 PDU without fragmentation.

To prevent the influence of routing protocol have on data performance, fixed (static) routing with external routing tables is used in the simulations. The use of fixed routing also enables us to control and fix the end-to-end paths at 4 hops from source to destination across different density networks so that fair comparisons can be made among them. The key parameters used in the simulations are summarised in Table 1.

5.2. Performance Metrics

The following two synchronization performance metrics in the SYNC windows are first evaluated.

1. Average waiting period for *sync* transmission (AWPST): This is the number of frames a node needs to wait from the time a *sync* packet is scheduled to the time it is transmitted.
2. Fraction of desired *sync* inter-arrival time (FDSIT): This is the fraction of *sync* packet received intervals that are smaller than N_{RP}, the desired *sync* received interval.

Table 1: Simulation Parameters

Parameter	Value	Parameter	Value
grid size	500m x 500m	data rate (CBR)	1 pkt/min
bandwidth	20 kbps	packet size	100B
simulation time	9000s	routing	fixed
tx power	36 mW	no. of hops	4
rx power	14 mW	retry limit	5
idle power	14 mW	N_{SP}	10 frames
sleep power	0 mW	N_{RP}	10 frames
CS range	550m	α	0.5
tx range	250m	C_{thres}	3

These two metrics are the direct outcomes of the different *sync* packet scheduling, transmission and reception mechanisms in the synchronization algorithms. Higher AWPST means that the sensor nodes have less sleep time in SYNC windows and hence have higher energy consumption for the synchronization process. Higher FDSIT means that the probability of sensor nodes getting out of synchronization is higher and data performance will be affected.

In addition, the energy and data performances of the sensor networks with different synchronization algorithms are also evaluated using the following three performance metrics:

1. Packet delivery ratio (PDR): This is the ratio of data packets delivered to the destination to total data packets generated by the source.
2. Average packet delay (APD): This is the end-to-end delay between the time the data packet is sent from the source and the time it is received by the destination.
3. Average node energy consumption (ANEC): This is computed by dividing the total energy consumed by the node by the total simulation time.
4. Individual Node Energy Consumption: While ANEC measures network-wide average energy consumption, it is also important for energy consumption to be evenly distributed among individual nodes in the network. Network lifetime will be impacted if some of the sensor nodes consume more energy than the others. These nodes will be depleted faster and will cause network segmentation.

5.3. C-Sync Parameters

The behaviour of C-Sync algorithm is controlled by two parameters: C_{thres} for counter-based sync transmission algorithm, and α for exponential smoothing *sync* reception algorithm.

5.3.1. Variation of Counter Threshold (C_{thres})

In the multi-hop broadcast scenarios, the choice of C_{thres} used in a counter-based algorithm affects the broadcast performance in two ways. When a small C_{thres} value is used, there are significantly fewer number of rebroadcasts. However, the reachability will be sacrificed in a sparse network. Increasing the threshold will increase reachability but also increase the number of rebroadcasts.

For the scenarios of the *sync* broadcast in multi-hop WSNs, the key considerations are synchronization performance, which translates to data performance, as well as energy consumption performance. Simulations for a subset of network scenarios are performed using different C_{thres} and their performances are examined. To study the effect of C_{thres}, we vary the values of C_{thres} from 1 to 4 in our network scenarios. Both 10% and 2% duty-cycle networks, with network densities ranging from 3x3 to 7x7 grids are simulated. A midpoint value of 40 ppm clock drift is used throughout. The results of PDR, APD and energy consumption are shown in Figure 6.

At 10% dc, the differences in all three performance metrics are not significant as shown. The maximum differences are in energy consumptions in the 4x4 grid network, which is in the range of 2.2%. At 2% dc, PDR performance for the case of $C_{thres} = 4$ is the lowest among all. It is also the worst performer for all three metrics in the high density 7x7 grid networks. For the case of $C_{thres} = 1$, it has the worst APD performance for most of the scenarios and the highest energy consumption in low density networks. Between $C_{thres} = 2$ and $C_{thres} = 3$, $C_{thres} = 2$ has better PDR performance in low density networks, while $C_{thres} = 3$ has better energy consumption and APD performances in most of the scenarios. We will therefore use $C_{thres} = 3$ in the subsequent simulations for comparison among the different synchronisation algorithms in this work.

5.3.2. Variation of Smoothing Factor (α)

The smoothing factor α in the exponential-smoothing algorithm represents the weighting applied to the most recent data. Values of α that are

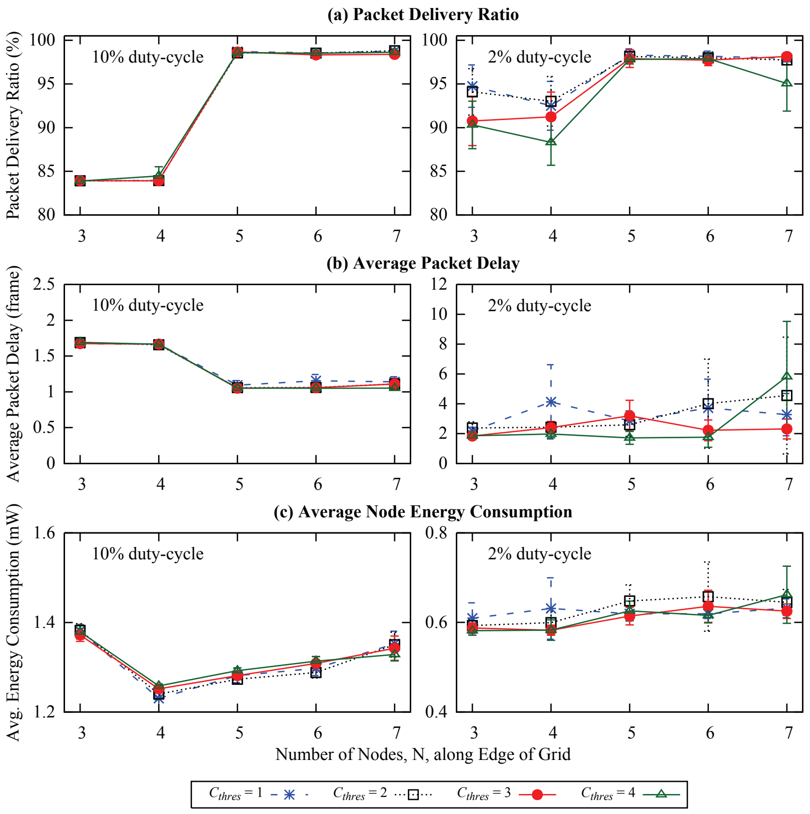

Figure 6: Energy and data performances of C-Sync algorithm with different C_{thres} in grid networks at 40 ppm drift rate

close to one have less of a smoothing effect and are more responsive to recent changes in the data, while the opposite is true for values of α closer to zero.

To study the effect of α, we use three different values of α at 0.25, 0.50 and 0.75 for the same grid network scenarios used in the previous section. The results of PDR, APD and energy consumption are shown in Figure 7.

The results of the simulations have shown that each of the three α values has its strength in different scenarios for different metrics. At 10% dc, PDR and APD for all three α are similar. In terms of energy consumption, the case of $\alpha = 0.75$ has the best performance in low density 3x3 grid networks while the case of $\alpha = 0.25$ has the best performance in high density 7x7 grid networks. At 2% dc, the case of $\alpha = 0.50$ has the best performances in all the three metrics measured in high density 7x7 networks. In 6x6 grid networks,

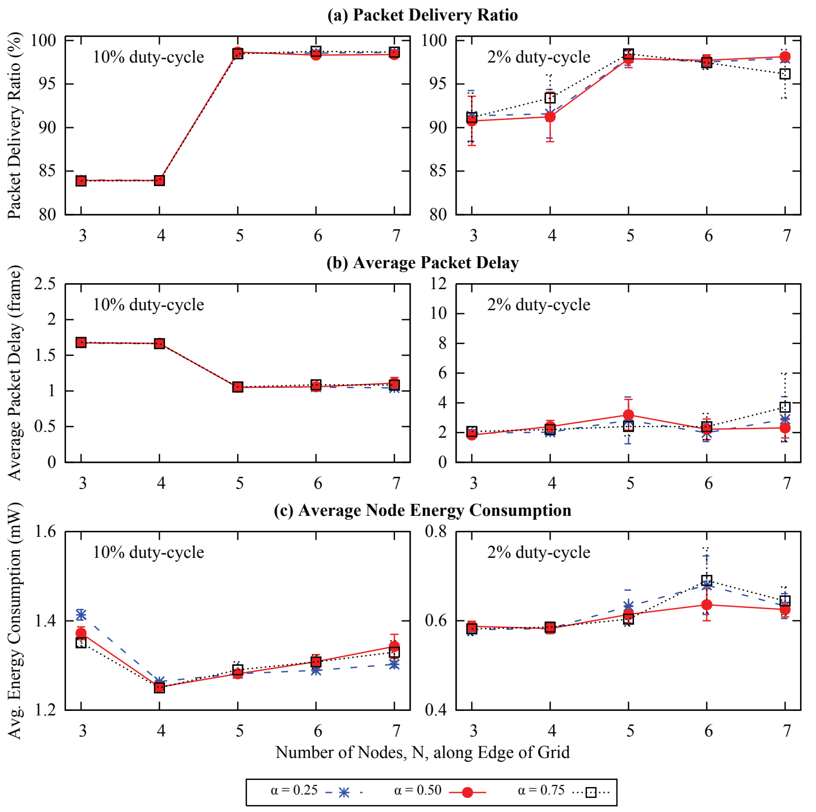

Figure 7: Energy and data performances of C-Sync algorithm with different α in grid networks at 40 ppm drift rate

it also has the best performance in energy consumption while maintaining similar performance on PDR and APD as the other two values of α.

5.4. Performance Results and Analysis

The three synchronization algorithms F-Sync, 1-Sync and C-Sync are simulated under different network densities, clock drift rates, and duty cycles. Means and 95% confidence intervals of the performance metrics for each scenario are plotted for evaluation.

5.4.1. Performance in SYNC windows

Figure 8 shows the average waiting period from the time a *sync* packet is scheduled to the time it is transmitted in different density networks. For

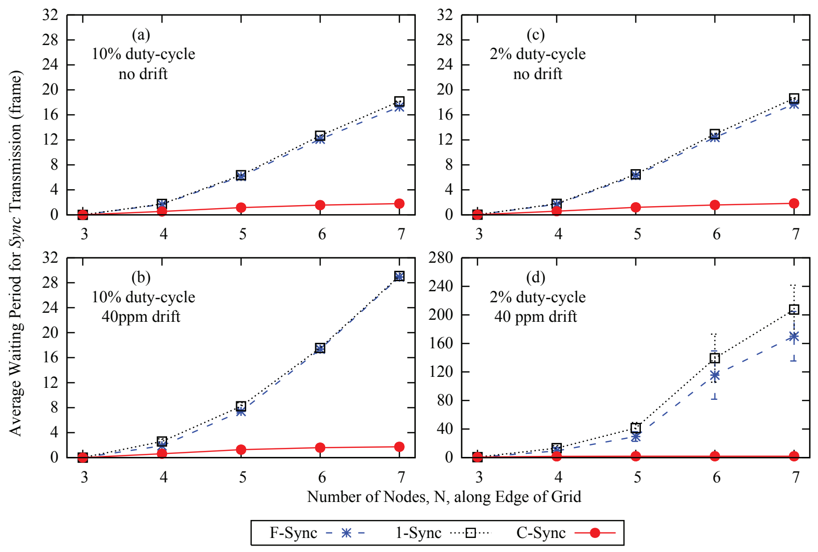

Figure 8: Average waiting period for *sync* transmission against different network densities

F-Sync and 1-Sync, AWPST increases due to the increased *sync* traffic and congestion as density increases. At 10% duty cycle with no drift, AWPST performance for F-Sync and 1-Sync are similar (Figure 8(a)), increasing from 0.0 frames in the 3x3 network to more than 17.0 frames in the 7x7 network. AWPST for C-Sync, on the other hand, are consistently less than 2.0 frames.

Comparing two different drift rates at 0 and 40 ppm, the reference time in the *sync* packet from a transmitting node deviates more from the receiving nodes' local clocks in general at a higher drift rate. If the time difference is greater than a threshold, the receiving nodes may misinterpret that the transmitting node is on a different sleep/wakeup schedule and this could trigger the receiving nodes to generate more *sync* packets. In high density networks, this will further increase the congestion and thus increase the AWPST.

In the scenario of the 7x7 network at 2% dc (Figure 8(c, d)), AWPST for F-Sync and 1-Sync increase tremendously from 17.7 and 18.6 frames at no drift to 170 and 207 frames at 40 ppm drift respectively. This means that F-Sync and 1-Sync nodes have little opportunity to sleep in SYNC windows and consume more energy. On the other hand, AWPST for C-Sync are consistently below 2.0 frames.

As can be seen from Figure 9, at the very low density of the 3x3 grid with 10% dc, 1-Sync and C-Sync achieve only 61.7% and 63.5% FDSIT respectively whereas F-Sync achieves almost 100% FDSIT because the nodes are active in all SYNC windows. When density increases, the performances of 1-Sync and C-Sync improve, achieving more than 99% FDSIT in the 7x7 grid network.

It is also worth noting that the presence of clock drift could cause the time difference between sensor nodes to be large. When a sensor node receives a *sync* packet from a sending node that has a time difference larger than a pre-specified threshold, the receiving node will assume that the sending node is on a different synchronisation schedule (multiple schedules). This will trigger the receiving node to schedule a new *sync* packet to be transmitted in the next SYNC window, which increases the number of *sync* packets transmitted in the network. The increase in the number of *sync* packets improves the FDSIT performance in the low density low duty-cycle networks. In the 3x3 grid network at 2% dc, FDSIT improves from 63% to 83% for 1-Sync, and from 64% to 95% for C-Sync.

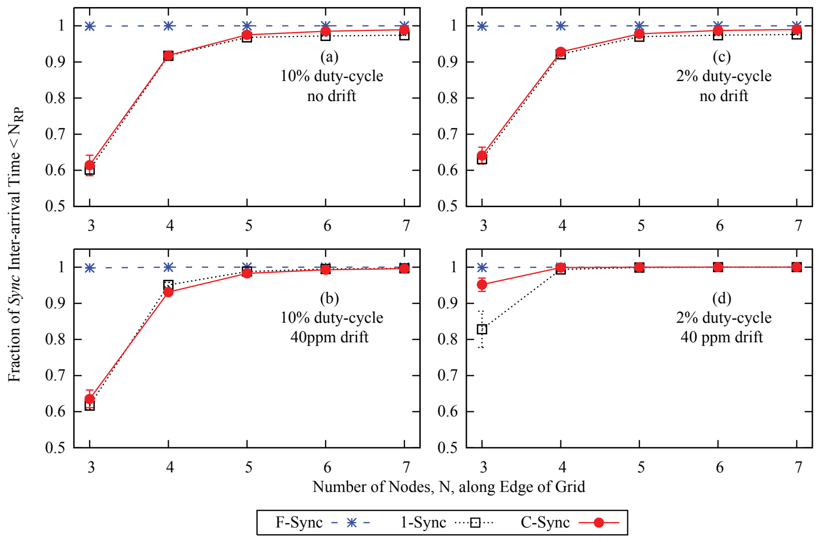

Figure 9: Fraction of *sync* inter-arrival time less than the predetermined period of NRP

5.4.2. Performance against Network Densities

The energy and data performances of the three synchronization algorithms for different density networks are shown in Figure 10.

As show in Figure 10(a), all three algorithms have the same PDR performance at 10% dc, and higher density networks have a better performance

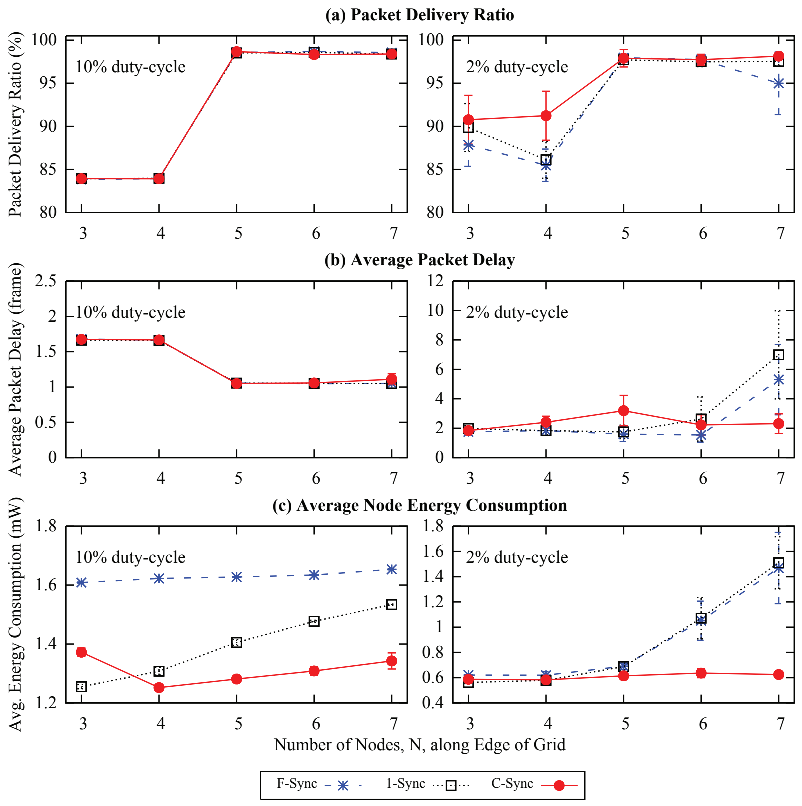

Figure 10: Energy and data performance for F-Sync, 1-Sync and C-Sync in different density grid networks at 40 ppm drift rate

than the lower density networks. At 2% dc, C-Sync has the best PDR performance among the 3 algorithms ranging from 90.8% to 98.1%.

Figure 10(b) shows the APD performance. Similarly, there is no significant difference in APD performance at 10% dc. At 2% dc, APD performance deteriorates for F-Sync and 1-Sync when density increases, reaching 5.3 and 7.0 frames respectively. C-Sync, on the other hand, has a consistent performance of APD less than 3.2 frames.

As shown in Figure 10(c), C-Sync is the most energy efficient algorithm except in a very sparse network of 3x3 grid at 10% dc. As network density increases, the relative efficiencies of C-Sync become better. For the 7x7 grid network at 2% dc, average energy consumptions for F-Sync and 1-Sync nodes are 135% and 141% higher than C-Sync nodes.

5.4.3. Performance against Duty Cycles

The energy and data performances of the three synchronisation algorithms in different duty-cycle networks are shown in Figure 11.

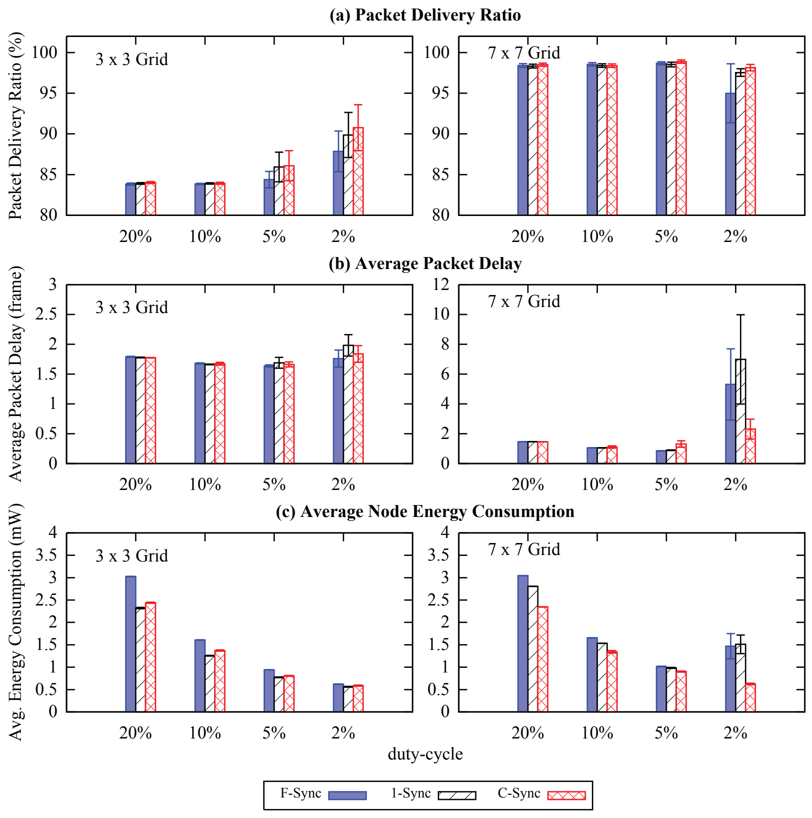

Figure 11: Energy and data performance for F-Sync, 1-Sync and C-Sync in 3x3 and 7x7 grid networks at 40 ppm drift at different duty cycles

Figure 11(a) shows the PDR performance. In both 3x3 and 7x7 grid networks, C-Sync has similar or better PDR performance than F-Sync and 1-Sync in all duty cycles.

As shown in Figure 11(b), there is no significant difference in the APD performance in the 3x3 grid network. However, in the 7x7 grid network at 2% dc, F-Sync and 1-Sync have 130% and 202% longer delay than C-Sync. At 2% dc, both F-Sync and 1-Sync also have significantly longer delays, as well as significantly larger delay variations, compared to their performances at higher duty cycles.

As shown in Figure 11(c), F-Sync has the highest energy consumption in all except one scenario. Only in the 7x7 grid network with 2% dc, does it have marginally lower consumption than 1-Sync. In the low density 3x3 grid network, 1-Sync has the best energy performance, consuming 4% to 9% lower energy than C-Sync. For the 7x7 grid network, C-Sync has the best energy performance, consuming 9% to 142% lower energy than 1-Sync.

Under normal circumstances, energy consumption in a low dc network is lower than in a high dc network due to the existence of longer sleep periods. However, the results show that energy consumptions of F-Sync and 1-Sync in the 7x7 grid network are higher at 2% dc than at 5% dc, which indicates that these two synchronisation algorithms may not be functioning well and are not suitable to operate in low dc, high density networks.

5.4.4. Performance against Clock Drifts

The energy and data performance of the 3 synchronisation algorithms in 3x3 and 7x7 grid networks with different clock drift rates are shown in Figure 12. The duty cycle for these scenarios is fixed at 2%.

As shown in Figure 12(a), in the dense 7x7 grid network, PDR decreases as drift increases, this is expected as synchronization becomes more challenging when clock drift increases, leading to more errors in data transmission. However, the opposite trend is seen in the sparse 3x3 grid network. When there is no drift, *sync* packets are few and far between in a sparse network. The presence of clock drift could trigger the generation of more *sync* packets in the network as explained in the previous section. The increase in the number of *sync* packets improves the synchronization performance (FDSIT performance shown in Figure 9), which also improves the PDR performance in the low density low duty-cycle 3x3 network. However, APD is not improved because it is measured based on the delivered *sync* packets only. Similarly energy consumption increases since more *sync* packets are transmitted with increasing drift.

Comparing the algorithms, C-Sync has the highest PDR ranging from 83.5% to 93.1% in the 3x3 grid network. In the 7x7 grid network, both C-Sync and 1-Sync have similar PDR ranging from 93.4% to 98.3%, which is marginally higher than F-Sync from 92.8% to 98.1%.

Figure 12(b) shows the APD performance. All 3 algorithms have a similar delay performance up to 20 ppm drift rate. The results of APD performance are inconclusive as the drift increases further.

In terms of energy consumption, there is no significant difference between

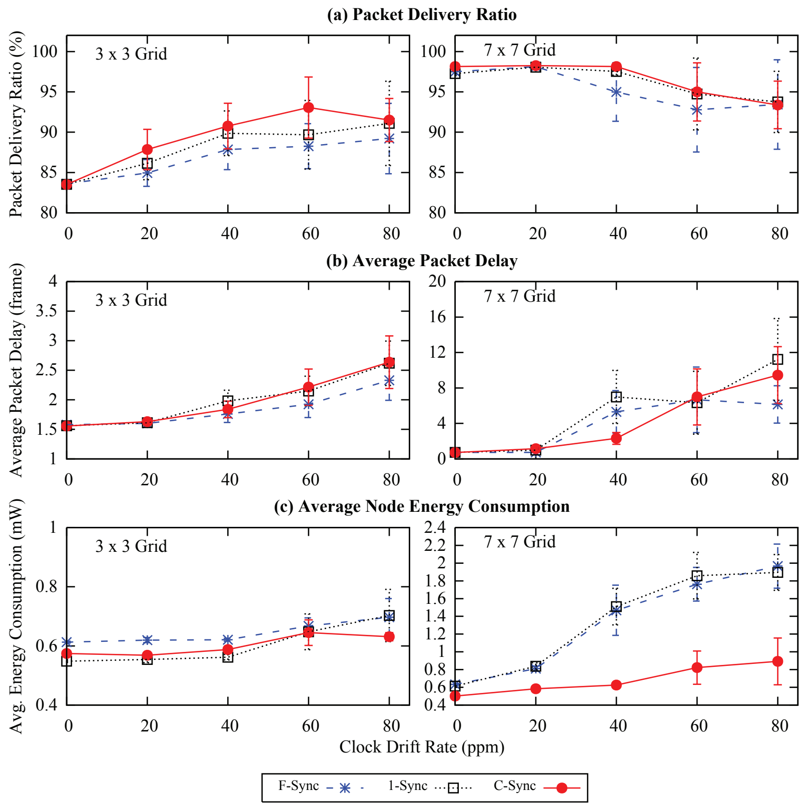

Figure 12: Energy and data performance for F-Sync, 1-Sync and C-Sync in 3x3 and 7x7 grid networks at 2% duty-cycle with different drift rates

1-Sync and C-Sync in the 3x3 grid network as shown in Figure 12(c), and both are up to 10.5% more efficient than F-Sync. The differences are more significant in the 7x7 grid scenarios; C-Sync is more energy efficient than F-Sync and 1-Sync in all drift scenarios. Energy savings range from 18.2% at zero drift to 69% at 80 ppm drift rate.

5.5. *Individual Node Energy Consumptions*

From the results in the previous sections, it is observed that average energy consumptions for F-Sync and 1-Sync increase significantly in high density (7x7 grid), low duty-cycle (2% dc), and high clock drift (80ppm) networks. To account for the unusually large increase in energy consumption, we look into the details of individual node energy consumptions within the grid networks in these scenarios.

5.5.1. Effect of Network Density

As shown in Figure 13 when the network density is low (4x4 grid), node energy consumption for all three algorithms are almost indistinguishable. The variations in energy consumption among different nodes are also small. As network density increases, both F-Sync and 1-Sync networks display large variations in node energy consumptions.

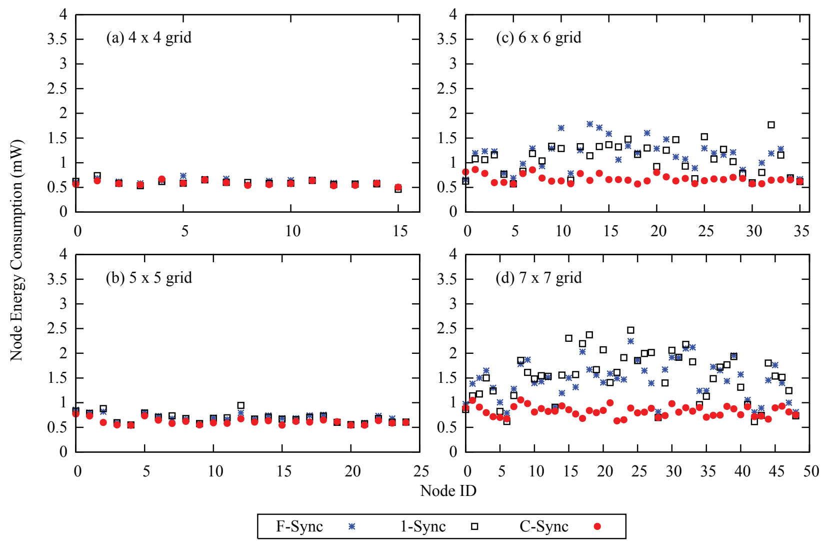

Figure 13: Individual node energy consumptions in different density grid networks at 2% duty-cycle with 40 ppm drift rate

The poor energy performances of F-Sync and 1-Sync in high density networks agree with the analysis of the *sync* congestion problem of the two algorithms in saturated networks. The large variations in node energy consumption will result in some nodes getting depleted of energy in much shorter times than the others, which will impact the connectivity and shorten the lifetime of the WSN. On the other hand, the counter-based *sync* transmission and *sync* cancellation sub-algorithm implemented in C-Sync has effectively removed this problem, as shown by the consistency in node energy consumption across different network densities.

5.5.2. Effect of Duty Cycle

Low duty-cycle operations appears to be a great challenge for F-Sync and 1-Sync as shown in Figure 14.

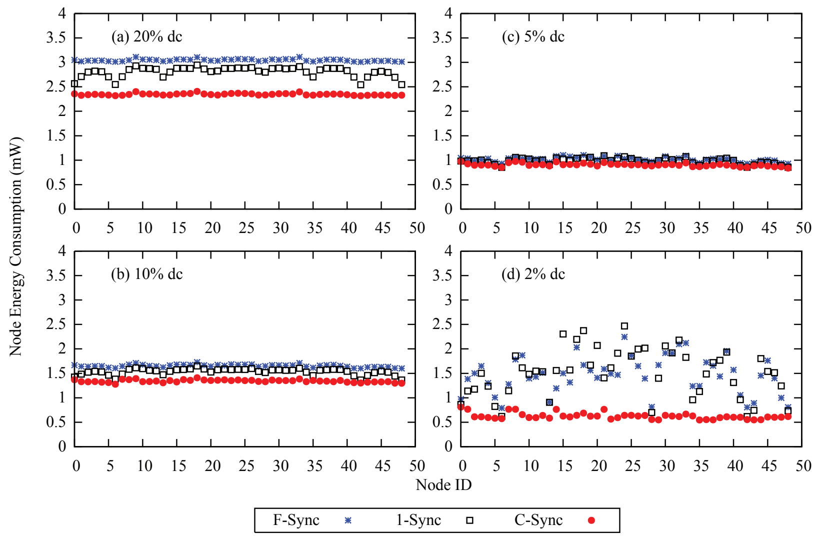

Figure 14: Individual node energy consumptions in 7x7 grid networks with 40 ppm drift rate at different duty-cycles

As shown in Figure 14, F-Sync and 1-Sync perform well in networks with duty cycles at 5% and higher. At 2% dc, both F-Sync and 1-Sync networks display large variations in node energy consumption. *Sync* packet inter-arrival times are comparatively longer in lower dc than in higher dc operations. At higher duty cycles with shorter *sync* intervals, a certain number of *sync* collisions and *sync* postponements can be tolerated. However, at lower duty cycles, *sync* collisions and *sync* postponements contribute to higher probabilities of asynchronous nodes, which cause instability in the networks and high variations in node energy consumption.

5.5.3. Effect of Clock Drift

As shown in Figure 15, when there is no drift, all three algorithms function well and energy consumptions are at similar levels among the different nodes in the network. As drift rate increases, not only do the energy consumptions of each node in F-Sync and 1-Sync increase significantly, the variability in node energy consumptions also increase significantly. C-Sync algorithm, on the other hand, maintains a consistent level of energy consumption among the different nodes in the network across all drift rates simulated.

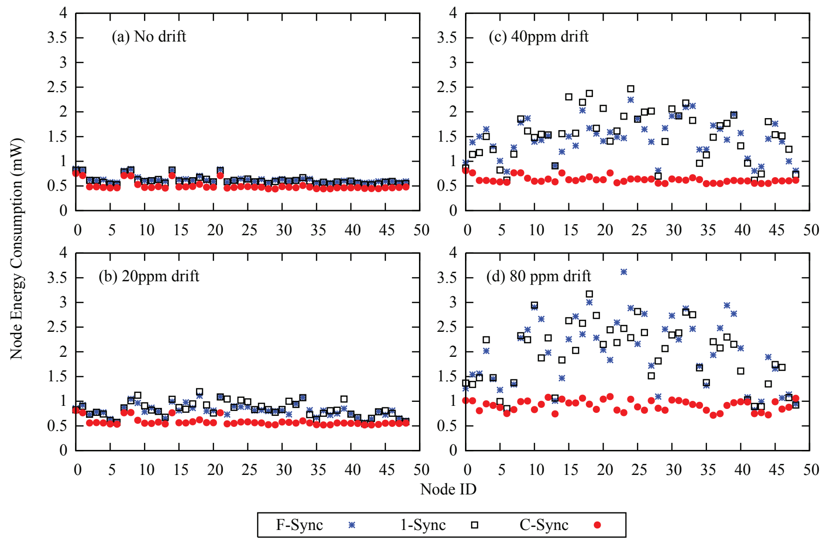

Figure 15: Individual node energy consumptions in 7x7 grid network at 2% duty-cycle with different drift rates

6. Conclusions

In this paper, we propose C-Sync, an adaptive synchronization algorithm for duty-cycle MAC protocols. C-Sync reduces energy consumption by adaptively regulating the synchronization traffic and synchronization wakeup period based on the changing network neighborhood conditions through counter-based and exponential-smoothing algorithms. The combination of counter-based *sync* transmission and exponential-smoothing *sync* reception algorithms effectively reduces the congestion and collision in the network when the *sync* traffic is high and maintains synchronization performance when the *sync* traffic is low.

From the results of the simulations, C-Sync consistently outperforms F-Sync and 1-Sync in terms of packet delivery ratio, average packet delay and energy consumption in most of the scenarios. The relative energy performance of C-Sync is also significantly better in the more challenging scenarios of high density, high drift and low duty-cycle networks.

References

[1] P. Huang, L. Xiao, S. Soltani, M. Mutka, N. Xi, The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey, IEEE Communications Surveys Tutorials 15 (2013) 101–120.

[2] M. R. Ahmad, E. Dutkiewicz, X. Huang, A Survey of low duty cycle MAC protocols in wireless sensor networks, in: Emerging Communications for Wireless Sensor Networks, InTech, Croatia, 2010, pp. 69–90.

[3] I. F. Akyildiz, M. C. Vuran, Wireless Sensor Networks, volume 4, John Wiley & Sons, 2010.

[4] W. Ye, J. Heidemann, D. Estrin, Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks, IEEE/ACM Transactions on Networking 12 (2004) 493–506.

[5] Y. Sun, S. Du, O. Gurewitz, D. B. Johnson, DW-MAC: a low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks, in: Proceedings of the 9th ACM international, Citeseer, 2008.

[6] Y. Z. Zhao, M. Ma, C. Y. Miao, T. N. Nguyen, An energy-efficient and low-latency MAC protocol with Adaptive Scheduling for multi-hop wireless sensor networks, Computer Communications 33 (2010) 1452–1461.

[7] Y. Z. Zhao, C. Y. Miao, M. Ma, An Energy-Efficient Self-Adaptive Duty Cycle MAC Protocol for Traffic-Dynamic Wireless Sensor Networks, Wireless Personal Communications 68 (2013) 1287–1315.

[8] K. Nguyen, Y. Ji, S. Yamada, Low overhead MAC protocol for low data rate wireless sensor networks, International Journal of Distributed Sensor Networks 2013 (2013).

[9] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, in: Proceedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, volume 3, pp. 1567–1576.

[10] Y.-C. Wu, Q. Chaudhari, E. Serpedin, Clock Synchronization of Wireless Sensor Networks, IEEE Signal Processing Magazine 28 (2011) 124–138.

[11] M. Maróti, B. Kusy, G. Simon, Á. Lédeczi, The Flooding Time Synchronization Protocol, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys '04, ACM, New York, NY, USA, 2004, pp. 39–49.

[12] J. Shannon, H. Melvin, A. Ruzzelli, Dynamic Flooding Time Synchronisation Protocol for WSNs, in: 2012 IEEE Global Communications Conference (GLOBECOM), pp. 365–371.

[13] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync Protocol for Sensor Networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03, ACM, New York, NY, USA, 2003, pp. 138–149.

[14] A. Ahmad, D. Zennaro, L. Vangelista, E. Serpedin, H. Nounou, M. Nounou, A distributed algorithm for network-wide clock synchronization in wireless sensor networks, in: 2013 16th International Conference on Information Fusion (FUSION), pp. 1037–1043.

[15] J. Elson, L. Girod, D. Estrin, Fine-grained network time synchronization using reference broadcasts, ACM SIGOPS Operating Systems Review 36 (2002) 147–163.

[16] F. Gong, M. L. Sichitiu, CESP: A power efficient, accurate coefficient exchange synchronization protocol, in: Wireless for Space and Extreme Environments (WiSEE), 2013 IEEE International Conference on, IEEE, 2013, pp. 1–6.

[17] S. Youn, A Comparison of Clock Synchronization in Wireless Sensor Networks, International Journal of Distributed Sensor Networks (2013).

[18] P. Sthapit, J.-Y. Pyun, Intelligent network synchronization for energy saving in low duty cycle MAC protocols, in: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops, pp. 1–6.

[19] K.-P. Ng, C. C. Tsimenidis, W. L. Woo, Energy-efficient synchronization algorithm for duty-cycle MAC protocols, in: 2015 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), IEEE, 2015, pp. 255–260.

[20] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, J.-P. Sheu, The Broadcast Storm Problem in a Mobile Ad Hoc Network, Wireless Networks 8 (2002) 153–167.

[21] D. Koscielnik, J. Stepien, The methods of broadcasting of information in ad-hoc wireless networks with mobile stations, in: 2011 IEEE International Symposium on Industrial Electronics (ISIE), pp. 2043–2048.

[22] Y.-C. Tseng, S.-Y. Ni, E.-Y. Shih, Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network, IEEE Transactions on Computers 52 (2003) 545–557.

[23] M. I. A. Khan, M. Atif, An Overview of Mobile Ad-hoc Network Simulators and Associated Simulation Techniques, International Journal of Computer Science and Telecommunications 3 (2012).