# A Localized De-synchronization Algorithm for Periodic Data Reporting in IEEE 802.15.4 WSNs

Domenico De Guglielmo
Dept. of Information Engineering
University of Pisa, Italy
d.deguglielmo@iet.unipi.it

Giuseppe Anastasi
Dept. of Information Engineering
University of Pisa, Italy
giuseppe.anastasi@iet.unipi.it

Marco Conti
IIT-CNR
National Research Council, Italy
marco.conti@iit.cnr.it

*Abstract* — **Energy efficiency is typically the most important requirement in wireless sensor networks (WSNs). However, in many application domains, additional requirements such as *reliability*, *timeliness* and *scalability* need to be considered as well. As emphasized in previous studies, the IEEE 802.15.4 MAC protocol has severe limitations in terms of performance and energy efficiency, which make it unsuitable for many application scenarios. In this paper we propose an asynchronous adaptive algorithm for periodic data reporting that leverages the 802.15.4 Beacon-Disabled Mode but de-synchronizes the access times of different sensor nodes, so as to minimize the collision probability. Unlike previous de-synchronization schemes, the proposed approach only relies on local information. Simulation results show that our asynchronous algorithm can provide a performance very similar to that of a TDMA scheme but, unlike TDMA, it does not requires any synchronization among nodes.**

***Keywords: IEEE 802.15.4, MAC Protocol, Beacon-Enabled Mode, Beacon-Disabled Mode, Energy Efficiency.***

## I. INTRODUCTION

*Energy efficiency* is usually the major concern in wireless sensor networks (WSNs) design. This is because sensor nodes are typically powered by batteries – with a limited energy budget – which cannot be replaced nor recharged, due to environmental or cost constraints [1]. However, in many real-life applications, additional requirements need to be taken into account, such as *scalability*, *reliability*, *timeliness* etc. Scalability is important as the number of deployed sensor nodes may be very high, especially when large geographical areas need to be monitored. Reliability and timeliness may be also very important issues, especially in critical application scenarios (e.g., process control, monitoring in industrial environments).

Recently, two industrial standards for WSNs have been released by the IEEE and ZigBee Alliance, respectively. The IEEE 802.15.4 standard [2] defines the physical and MAC (Medium Access Control) layers of the protocol stack, while the ZigBee specifications [3] cover the networking and application layers. Products compliant to the above-mentioned standards are largely available on the market and they are considered the reference technology for WSNs. However, a number of studies have emphasized that the IEEE 802.15.4 MAC protocol has several limitations, in terms of scalability, reliability, predictability in latency experienced by packets, and also energy efficiency [4, 5, 6, 7, 8].

Due to these limitations, the 802.15.4 MAC protocol is unsuitable for critical applications with stringent reliability and/or delay constraints. In such scenarios, Time Division Multiple Access (TDMA) is typically used for communication among sensor nodes. As well known, when using TDMA, time is divided into transmission slots that are pre-assigned to sensor nodes. Each sensor node activates only during its own slot(s) and sleeps for the rest of the time, thus saving energy. Therefore, TDMA provides guaranteed bandwidth, high energy efficiency, absence of collisions (i.e., high reliability), low and predictable latency. On the other side, it requires a strict *synchronization* among sensor nodes, and has a *limited flexibility* since a change in the network topology may require a different slot allocation.

To overcome TDMA limitations, de-synchronization schemes have been recently proposed for periodic data reporting in single-hop [9, 10] and multi-hop [11, 12, 13] WSNs. As suggested by the name, *de-synchronization* is the opposite of *synchronization* and implies that each sensor node performs its periodic data transmissions as far away as possible from all other nodes [9]. De-synchronization schemes can thus be used to arrange periodic transmissions from different sensor nodes in an interleaved, round-robin style, like in conventional TDMA. Unlike TDMA, however, de-synchronization schemes do not require a strict synchronization among sensor nodes.

In this paper we propose an Asynchronous Adaptive Periodic (AsAP) algorithm for desynchronizing periodic transmissions of sensor nodes. The proposed solution is tailored to the (unslotted) 802.15.4 CSMA/CA algorithm, but it can be potentially extended to work upon any random-access MAC protocol for WSNs. Unlike previous de-synchronization schemes, where sensor nodes adapt their behavior on the basis of information received from other nodes [9] or from a coordinator node [12], our algorithm is completely decentralized and sensor nodes adapt their behavior in a *fully autonomous* way, only relying on *local* information. This makes the algorithm robust and suitable for environments where packets can be corrupted or missed. In addition, nodes are not required to be always active to look for information from neighboring nodes. We evaluated the proposed algorithm via simulation. The obtained results show that AsAP is scalable and provides performance very similar to that of a TDMA scheme, in terms of energy consumption, latency, and reliability.

The rest of the paper is organized as follows. Section II presents the AsAP algorithm. Section III describes the simulation setup used for our analysis, while section IV discusses the performance of the proposed algorithm. Finally, conclusions are drawn in Section V.

## II. ASAP ALGORITHM DESCRIPTION

In this section we present our AsAP algorithm. It is an heuristic algorithm whose design leverages the following principles.

- *Energy efficiency*. Since energy is a limited resource in WSNs, this is the main target of the algorithm.
- *Scalability*. The algorithm should exhibit performance similar to that of TDMA for a number of sensor nodes similar to that allowed by TDMA.
- *Flexibility*. The algorithm should manage efficiently possible changes in the network topology.
- *Locality*. The algorithm should rely only on local information available at the sensor node. No information exchange is required for coordination with other nodes or coordinator node.
- *Simplicity*. The algorithm should be as simple as possible in order to be used in sensor nodes with limited computational and energy resources.

The basic idea behind our algorithm is very simple. Each sensor node looks for a position (over the operation period *T*), free of collisions with other nodes, only basing on local information. Initially, the sensor node generates a random transmission time and, then, adjusts its transmission time depending on the outcome of the previous transmission, i.e., failure or success and, in case of failure, the reason for that (i.e. exceeded number of re-transmissions or exceeded number of backoff stages).

We now describe the AsAP algorithm in detail, assuming that it is running on top of the 802.15.4 MAC protocol (unslotted version) [2]. However, the algorithm can be easily extended to any contention-based MAC protocol. We assume that all sensor nodes generate packets periodically, with a common fixed period *T*. Algorithm 1 shows the specific actions performed by each sensor node. In Algorithm 1, *macMinBE* denotes the 802.15.4 MAC parameter used to compute the initial backoff-window size [2]. Specifically, each node, before transmitting a packet, waits for a random backoff time initially generated in the range $[0, 2^{macMinBE}-1]\times 320\mu s$. Furthermore, $next\_send\_time \in [0,Ta]$ denotes the time selected by the algorithm for trasmitting in the next period. $T_a$ is equal to $T$ minus the time required to transmit a frame if the maximum backoff has been extracted, i.e. $T_a=T - D_{frame\text{-}max}$. Hereafter, all the times are taken with respect to the start time of the current period.

**Algorithm 1**: AsAP Algorithm

```
macMinBE = 3; Pc=0.5;
failure_count=0; failure_threshold=3;
choose τ in [0,Ta];
next_send_time= τ ;
Loop
sleep until time=next_send_time;
send packet;
wait(notification);
switch(notification) {
case(tx-success AND no-retransmissions)
  next_send_time =
      = (tsuccess-Dack-Dtat-Dframe-Dtat-DCCA-Didle-rx) mod Ta;
  macMinBE = 0;
case (tx-success AND retransmissions)
  next_send_time = next_send_time;
case (tx-failure  AND exceeded-number-of-backoffs)
  next_send_time = tfailure mod Ta;  macMinBE=3;
case (tx-failure AND exceeded-number-of-rtx)
  failure_count++;
  If (failure_count < failure_threshold)
       next_send_time = next_send_time;
  else
    Pc:        choose τ in [0,Ta]; next_send_time= τ ;
               macMinBE=3;
    (1-Pc):    next_send_time = next_send_time;
  end if
}
end Loop
```

Initially, *macMinBE* is set to the default value 3, and a random transmission time, uniformly distributed in $[0,Ta]$ is selected. Then, at each period, the sensor node wakes up at the time scheduled for transmission, sends a packet to the MAC protocol below, and waits for the related notification. The following four different outcomes can occur: **(i)** the packet is successfully received by the sink after the first attempt; **(ii)** the packet is successfully received after one or more re-transmissions; **(iii)** the packet is discarded due to exceeded number of backoff stages; **(iv)** the packet is discarded due to exceeded number of re-transmissions. The different cases will be discussed below.

If the packet is successfully transmitted after the first attempt (case (**i**)), it means that no collisions have occurred and, hence, the selected portion of the period is (apparently) free of competitors. Therefore, the same portion will be re-used in the next period. However, to minimize latency and energy consumption, the preliminary phase due to the random backoff time will be avoided. Therefore, the send primitive for the next period is scheduled at a time corresponding to the actual starting point of the current transmission and *macMinBE* is set to 0. To derive the time for the next send operation let us denote by $t_{success}$ the time when the success notification is received from the underlying MAC protocol. Assuming that propagation and processing delays are negligible, this time is approximately the time when the acknowledgement has been received by the MAC layer. Hence, with reference to Figure 1, the time for the next send operation can be calculated as:

$$next\_send\_time = \left(t_{success} - D_{ack} - 2D_{tat} - D_{frame} - D_{CCA} - D_{idle-rx}\right) \bmod T_a \quad (1)$$

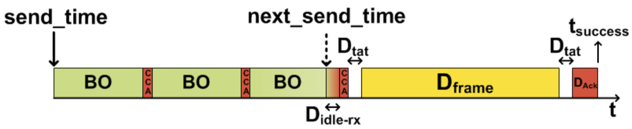



Figure 1. Derivation of the send time for the next period when the packet is transmitted successfully after the first attempt.

where $D_{frame}$ ($D_{ack}$) is the time required for transmitting a data (ack) frame, $D_{tat}$ is the turnaround time, $D_{cca}$ is the time needed to perform a CCA, and $D_{idle\_rx}$ the time taken to switch the radio from idle to receive mode.

Let us consider now case (**iii**), when the packet is discarded due to exceeded number of backoff stages. This implies that the portion of time (randomly) selected by the sensor node is very likely congested and it is, thus, convenient not to retry the same time in the next period. Instead, the sensor node will start sending at a time corresponding to the end of the current transmission. Hence, denoting by $t_{failure}$ the time when the failure notification is received from the MAC layer, the next send time is derived as:

$$next\_send\_time = t_{failure} \bmod T_a \quad (2)$$

Since we cannot be sure that the time interval considered for the next period is free of competitors, the *macMinBE* parameter is set to the default value (i.e. 3). We verified by simulation that selecting the next transmission time according to (2), instead of randomly choosing a new time instant is beneficial in terms of performance, scalability and convergence to a steady state.

When the packet is discarded due to exceeded number of retransmissions (case (**iv**)) it means that the node found the channel idle and procedeed with frame transmission several times. Unfortunately, the packet was not received correctly by the sink due to channel errors and/or collisions. If the packet was corrupted due to channel errors, it makes no sense to change the send time since that the channel unreliability is tipically a transient phenomenon. Instead, if retransmissions are caused by collisions, there is a (hidden) conflicting node and, hence, a change in the send time is convenient. Obviously, it is not possible to discriminate the two cases (i.e. conflicting node vs. channel error) only relying on local information. We always assume a channel error unless a number of consecutive failures, equal to *failure_threshold,* is observed. In the latter case we assume there is a hidden conflicting node. Hence, to avoid a new collision in the next period, the sensor selects a new randomly generated transmission time with probability $P_c$. Instead, it retries the same transmission time with probability 1-$P_c$. Lines (18-25) show the *next_send_time* computation.

In the last case (case (**ii**)), i.e. when the packet is correctly received after one or more re-transmissions, the next send time is exactly the same as in the current period. This is because retransmissions are tipically due to channel errors rather than to collisions, and the channel unreliability is tipically a transient phenomenon as mentioned above.

## III. SIMULATION SETUP

To perform our simulation analysis we implemented the AsAP algorithm using the ns2 simulation tool [14], which includes the 802.15.4 module. We compared the performance achieved when using our AsAP algorithm with that provided by the Beacon Enabled (BE) and Beacon Disabled (BD) access modes of the 802.15.4 MAC protocol [2]. In our analysis we considered a star network scenario where the sink node acts as the network coordinator and sensor nodes are placed in a circle centered at the sink node, 10 m far from it. The transmission range was set to 15 m, while the carrier sensing range was set to 30 m (according to the model in [15]). We considered a periodic reporting application where sensed data have to be reported to the sink periodically. Time is divided into communication periods of duration $T$ and each sensor node generates one or more data packets every $T$ seconds. The frame size is the maximum allowed by the 802.15.4 (127 bytes) while the size of an ack frame is 11 bytes. All other parameters used are listed in Table 1. Unless stated otherwise the values shown in Table 1 have been used. The acknowledgement mechanism is always enabled.

In our analysis we compared the performance of AsAP with that of the following access methods which are commonly used in practice.

- *802.15.4 Beacon Enabled (BE) Mode*. Each sensor node is assumed to generate data just before the reception of the periodic Beacon. Therefore *all* sensor nodes compete for channel access at the beginning of *each* Beacon period. i.e., access times are synchronized. This access method maximizes competition.
- *802.15.4 Beacon Disabled (BD) Mode*. Sensor nodes are assumed to generate and transmit data at random times within the period $T$. Each sensor node initially picks a random transmission time $\tau_0$, uniformly distributed in [0,$T_a$]. Subsequent transmission times will occur at time $\tau_n = \tau_0 + nT$, where $n$=0,1,2,… denotes the $n$-th period. This scheme tries to minimize contention, however conflicts can still occurs.
- *TDMA*. Transmission slots are pre-assigned and each sensor node is active only during its allocated time slot(s). Therefore, TDMA avoids conflicts and minimizes access latency and energy consumption.

When using the 802.15.4 BE mode the Beacon period and the Active period are expressed through the *Beacon Order* (BO) and *Superframe Order* (SO) 802.15.4 MAC parameter, respectively [2]. In our experiments we set BO=SO=6, corresponding to a period $T$ of approximately 1s (0.983s). To make the comparison fair we used the same $T$ value for all the other considered schemes.

TABLE 1. PARAMETERS USED IN OUR ANALYSIS

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $T$ | 0.983 s | $T_a$ | 0.9756 s |
| *macMaxFrameRetries* | 0 | $D_{frame}$ | 4.256 ms |
| *macMaxCSMABackoffs* | 4 | $D_{ack}$ | 0.352 ms |
| *macMaxBE* | 5 | $D_{idle\text{-}rx}$ | 0.192 ms |
| *macMinBE* | 3 | $D_{idle\text{-}tx}$ | 0.192 ms |
| $P_{rx}$ | 35.46 mW | $D_{tat}$ | 0.192 ms |
| $P_{tx}$ | 31.32 mW | $D_{CCA}$ | 0.128 ms |
| $P_{idle}$ | 0.7668 mW | *LIFS* | 0.640 ms |
| $P_s$ | 0.036µW | $D_{frame\text{-}max}$ | 7.36ms |

### A. Performance Indices

In our analysis we considered the following indices.

- *Delivery ratio*, defined as the ratio between the number of data packets correctly received by the sink and the total number of data packets generated by *all* sensor nodes. It measures the network *reliability* in the data collection process.
- *Average latency*, defined as the average time from when the packet transmission is started at the source node to when the same packet is correctly received by the sink. It characterizes the *timeliness* of the system.
- *Average energy per packet*, defined as the total energy consumed by each sensor node divided by the number of data packets correctly delivered to the sink. It measures the *energy efficiency* of the WSN.

The energy consumed by a sensor node was calculated using the model presented in [16], which is based on the Chipcon CC2420 radio transceiver [17]. Specifically, the model supports the following radio states: *transmit*, *receive*, *idle* (the transceiver is on, but it is not transmitting nor receiving, i.e., it is monitoring the channel) and *sleep* (the transceiver is off and can be switched back on quickly). In addition, the model accounts for the energy spent due to state transitions as well. Although the standard does not explicitly state when the transceiver should be sleeping – except for the inactive portion of the superframe when the beacon-enabled mode is used – to further improve the energy efficiency we put the transceiver into the sleep state when there is no packet to be transmitted, as in [8]. In our experiments, for each simulated scenario, we performed 10 independent replications, where each replication consists of 1000 periods. For each replica we discarded the initial transient interval (10% of the overall duration) during which nodes associate to the coordinator node and start generating data packets. The results shown below are averaged over all the different replications. We also derived confidence intervals by using the independent replication method. However, they are tipically so small that they cannot be appreciated in the figures below.

## IV. PERFORMANCE EVALUATION

### A. Preliminary Analysis

In this section we derive, analytically, the performance figures of TDMA that will be used in the next section as reference bounds to compare the performance of our AsAP algorithm. Figure 2 shows a typical frame transmission using TDMA over the IEEE 802.15.4 physical layer. Since the radio is initially in idle mode it takes $D_{idle\text{-}tx}$ to transit to the transmit mode. Then, the frame transmission follows. After transmitting a data frame, the radio is switched to receive mode, which takes another turnaround time and, finally, the acknowledgement frame is received. With reference to Figure 2 and using parameter values in Table 1 the performance figures of TDMA can be easily calculated. The TDMA slot duration is $D_{TDMA} = D_{idle-tx} + D_{frame} + D_{tat} + D_{ack} = 4.992\,ms$, which implies that the maximum number of slots that can be accommodated in a period $T$ of 0.983 s is equal to 196 (corresponding to a maximum net throughput of approximately 188 Kbps). Assuming an ideal channel, the delivery ratio will be equal to 1 for a number of sensor nodes less than or equal to 196. The (constant) latency experienced by packets is equal to $L_{TDMA} = D_{idle-tx} + D_{frame} = 4.448\,ms$, while the total energy consumed by a sensor node for transmitting a data packet can be calculated as follows.

$$E_{TDMA} = D_{idle-tx} \cdot \frac{1}{2}\left(P_{idle} + P_{tx}\right) + D_{frame} \cdot P_{tx} + \\ + D_{tat} \cdot \frac{1}{2}\left(P_{rx} + P_{tx}\right) + D_{ack} \cdot P_{rx} = 0.155\,mJ$$

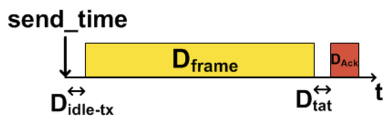
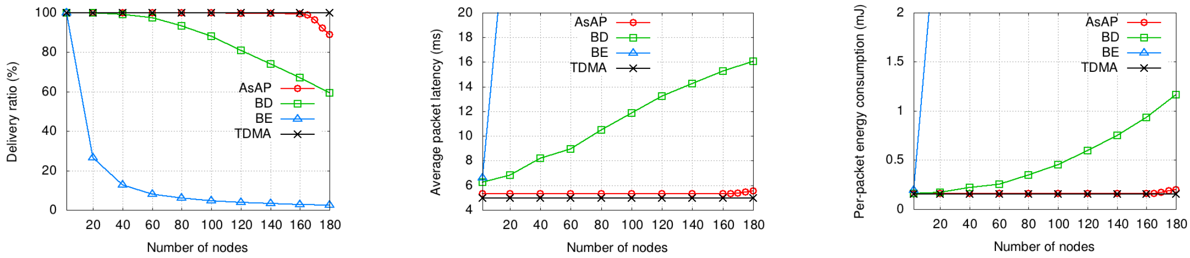



Figure 2. Data transmission with TDMA over the IEEE 802.15.4 Physical Layer

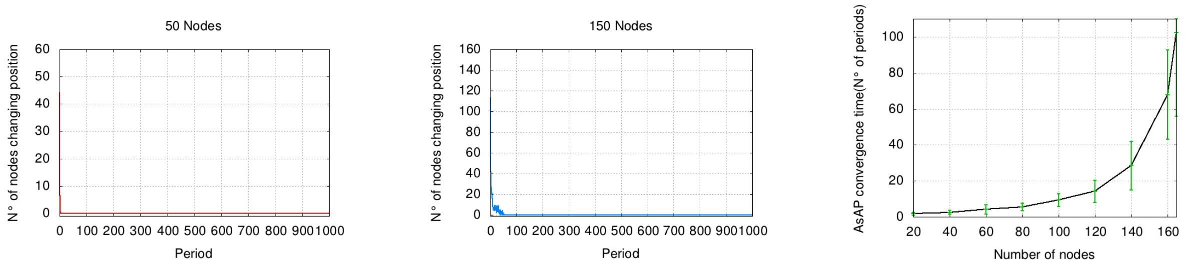
### B. Analysis in stationary conditions

In this section we analyze the performance of the different algorithms in stationary conditions. Specifically, we assume that, for each experiment, the number of sensor nodes is fixed and each node generates one packet per period $T$. Figure 3 shows the performance indices of the different algorithms for an increasing number of sensor nodes. As expected, AsAP outperforms both BE and BD and exhibits performance figures very close to those of TDMA, for all the considered indices. The delivery ratio provided by AsAP is very close to 1 even for a large number of sensor nodes and starts decreasing only when this number exceeds 165 (Figure 3-a). In Section 4.A we found that, for the same period $T$, the maximum number of slots (i.e., nodes) that can be accommodated is 196. However, when using AsAP the sensor node cannot access the wireless medium straightforwardly as in ideal TDMA. Instead it performs a preliminary CCA to assess the medium status and then, if the medium is idle, it transits from receive to transmit mode and starts transmitting the frame. Thus, for each transmission with AsAP we need to consider the additional delays due to CCA and turnaround time. By introducing these additional components in the expression of $D_{TDMA}$ we find that the maximum number of slots that can fit in the considered period $T$ reduces to 165. The same remarks can also apply to explain the small discrepancies in latency and energy introduced by AsAP and TDMA (Figure 3-b and Figure 3-c). The additional latency (energy) introduced by AsAP, with respect to TDMA, when the number of sensor nodes is less than or equal to 165 is approximately 7.2% (7%). We also investigated the convergence of the proposed algorithm, i.e., the number of periods required by the algorithm to reach a steady state, after which no sensor node changes its transmission time. Figure 4-a and Figure 4-b show, for two different network sizes, the number of sensor nodes that change their transmission time from a period to the next one. Apart from the first period, the number of such nodes is low and, after a certain number of periods, the network converges to a steady state condition. The convergence time (expressed in number of periods) for an increasing network size is shown in Figure 4-c. Even when the number of nodes in the network is very large, e.g., 160, the convergence time is about 70 periods.




(a) (b) (c)

Figure 3. Delivery ratio (a), average packet latency (b), and average energy consumed per packet vs. number of nodes (c).




(a) (b) (c)

Figure 4. Number of nodes changing access time in different communication periods with 50 (a) and 150 (b) active nodes and AsAP convergence time (c)

## C. Analysis in dynamic conditions

In this section we analyze the behavior of the algorithm in dynamic conditions. Specifically, we considered a network where the number of active nodes (and, hence, the offered load) changes over time. Initially, there are 50 sensor nodes. Then, at period 200, 50 more sensor nodes become active and, at period 400, a further third set of 50 nodes activates, thus increasing the total number from 50 to 100 and to 150, respectively. Then, at period 600, this number reduces to 100 and, finally, it reduces again to 50 (i.e., the initial value) at period 800. All sensor nodes, when active, generates one packet per period. Figure 5 shows the delivery ratio (a), average packet latency (b), average energy per transmitted packet (c) during the experiment. The value of these indices remains constant over the experiment, apart from a limited number of periods just after an increase in the number of sensor nodes. As expected, no variation is observed when the number of active nodes decreases. The convergence time, after an increase, is approximately the same as that observed in static conditions. Figure (d-f) emphasize that, when the number of active nodes increases, nodes already active are less affected by the transient than newly arrived nodes. All three figures show the number of nodes changing their transmission time from one period to the next. However, Figure 5-d refers to nodes that are active since the beginning, while Figure 5-e and Figure 5-f show the behavior of nodes that activate at period 200 and 400, respectively. We can see that during both transient intervals, the portion of already active nodes that are affected by the transient is significantly lower than that of newly arrived nodes. This is because newly arrived nodes initially use a *macMinBE* value equal to 3 (i.e., they perform a backoff before channel access), while already active nodes use *macMinBE*=0 (i.e., no backoff) which gives them priority.

## V. CONCLUSIONS

In this paper, we proposed an Asynchronous Adaptive Periodic (AsAP) algorithm for desynchronizing periodic transmissions of sensor nodes so as to obtain a free-of-collision schedule of transmissions, like in TDMA. The solution proposed here is tailored to the (unslotted)

802.15.4 CSMA/CA algorithm. Unlike previous de-synchronization schemes, where sensor nodes adjust their behavior exploiting information received from other nodes in the network, our algorithm is completely decentralized and sensor nodes operate in a *fully autonomous* way, only relying on *local* information. This makes the algorithm robust and suitable for environments where packets can be corrupted or missed. In addition, sensor nodes are not required to be always active to look for information from neighboring nodes. We have evaluated our algorithm, via simulation, both in stationary and dynamic conditions. The obtained results show that AsAP is scalable and provides performance very similar to that of an ideal TDMA scheme in terms of delivery ratio, latency and energy consumption. We have also implemented our algorithm in real Tmote Sky sensor network with TinyOS operating system. The experimental results, omitted for the sake of space, confirm the simulation results shown in Section 4.

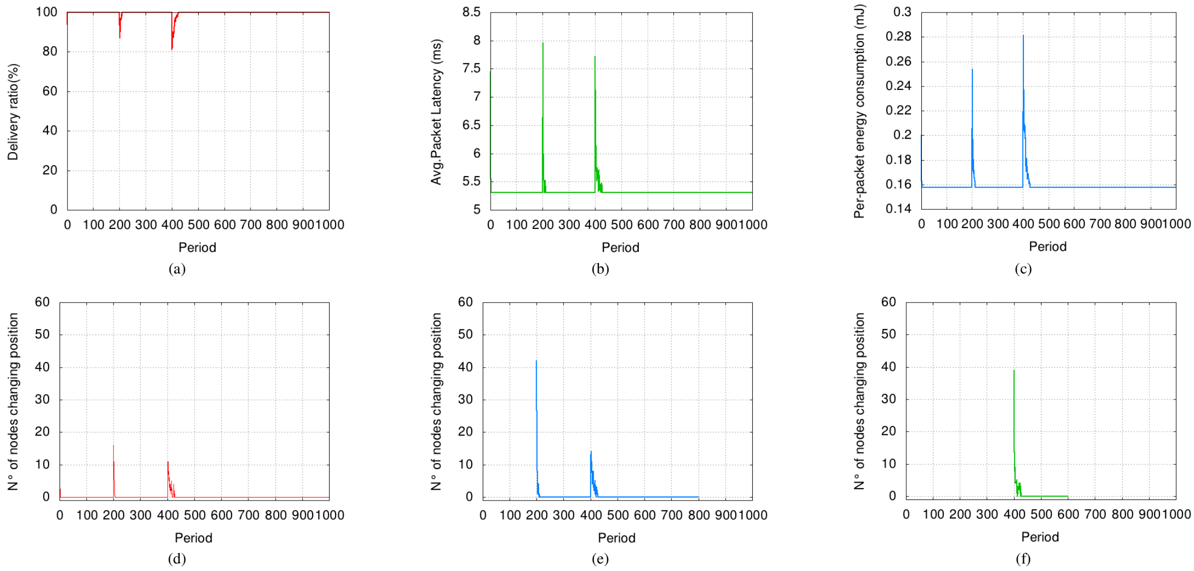



Figure 5. Delivery ratio (a), average packet latency (b), and average energy consumed per packet (c) experienced in different communication periods. Number of nodes changing their access time: nodes active since the beginning (d), nodes starting at period 200 (e), nodes activating at period 400 (f).

## REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, “Wireless Sensor Networks: a Survey”, *Computer Networks*, Vol.38, N. 4, March 2002.

[2] IEEE Standard for Information technology, Part 15.4; Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Computer Society, 2006.

[3] ZigBee Alliance, The ZigBee Specification version 1.0 (Q4/2007).

[4] K. Yedavalli, B. Krishnamachari, “Enhancement of the IEEE 802.15.4 MAC Protocol for Scalable Data Collection in Dense Sensor Networks”, *Proc. International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt08),* Berlin, Germany, March 31 - April 4, 2008.

[5] C. K. Singh, A. Kumar, P. M. Ameer, “Performance Evaluation of an IEEE 802.15.4 Sensor Network With a Star Topology”, *Wireless Networks*, Vol. 14, N. 4, August 2008.

[6] S. Pollin, M. Ergen, S. Ergen, B. Bougard, L. Van der Perre, I. Moerman, A. Bahai, F. Catthoor, “Performance Analysis of Slotted Carrier Sense IEEE 802.15.4 Medium Access”, *IEEE Trans. Wireless Communications*, Vol. 7, N. 9, September 2008.

[7] P. Park, P. Di Marco, P. Soldati, C. Fischione, K. H. Johansson, “A Generalized Markov Model for an Effective Analysis of Slotted IEEE 802.15.4”, Proc. *IEEE International Conference on Mobile Ad-hoc and Sensor Systems 2009 (IEEE MASS 09)*, Macau, China, October 2009.

[8] G. Anastasi, M. Conti, M. Di Francesco, "A Comprehensive Analysis of the MAC Unreliability Problem in 802.15.4 Wireless Sensor Networks", *IEEE Transactions on Industrial Informatics*, Vol.7, N.1, Feb. 2011.

[9] J. Degesys, I. Rose, A. Patel, R. Nagpal, “DESYNC: Self-Organizing Desynchronization and TDMA on Wireless Sensor Networks”, Proc. *Int’l Conference on Information Processing in Sensor Networks*, Cambridge, USA, April 25-27, 2007.

[10] R. Pagliari, Y. Hong, A. Scaglione, “Bio-Inspired Algorithms for Decentralized Round-Robin and Proportional Fair Scheduling”, *IEEE Journal on Selected Areas in Communications (J-SAC)*, Vol. 28, N. 4, May 2010.

[11] J. Degesys, R. Nagpal, “Towards Desynchronization of Multi-hop Topologies”, IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008), Venice, Italy, October 20-24, 2008.

[12] H. Kang, J. Wong, “A Localized Multi-Hop Desynchronization Algorithm for Wireless Sensor Networks”, Proceedings of IEEE INFOCOM 2009, Rio de Janeiro, Brazil, April 19-25, 2009.

[13] C. Mühlberger, R. Kolla, “Extended Desynchronization for Multi-Hop Topologies” *Technical Report*, Institut für Informatik, Universität Würzburg, 2009. http://www5.informatik.uniwuerzburg.de/publications/techreports/tr_extdesync.pdf.

[14] Network Simulator Ns2, http://www.isu.edu/nsnam/ns.

[15] G. Anastasi, E. Borgia, M. Conti, E. Gregori and A. Passarella, “Understanding the Real Behavior of 802.11 and Mote Ad hoc Networks”, *Pervasive and Mobile Computing*, Vol. 1, N. 2, 2005.

[16] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene, “Energy Efficiency of the IEEE 802.15.4 Standard in Dense Wireless Microsensor Networks: Modeling and Improvement Perspectives”, *Proc. Conference on Design, Automation and Test in Europe (DATE),* Vol.1, March 7-11, 2005.

[17] Chipcon CC2420 Website, http://focus.ti.com/docs/prod/folders/print/cc2420.html.