SESAM: A Semi–Synchronous, Energy Savvy, Application–Aware MAC

Renato Lo Cigno, Matteo Nardelli
DISI, University of Trento
Trento, Italy

Michael Welzl
Institute of Computer Science, University of Innsbruck
Innsbruck, Austria

***Abstract*—In spite of the huge recent research effort in the field, energy efficiency remains one of the key issues in wireless communications. The area most affected by energy inefficiency is Wireless Sensor Networks (WSN). In this paper we propose SESAM — a distributed MAC protocol, which, making use of application level information to predict future transmission instants between nodes, achieves an exceptional level of energy conservation.**

The protocol is most suited for low-bit-rate applications, or, more precisely, for applications and networking scenarios where the per-node transmission channel utilization is low. These are the only conditions under which energy consumption is a concern: in other conditions energy consumption is not dominated by transmission and reception of useful data.

A simple energy consumption model, supported by initial simulation results, shows very encouraging results, with energy consumption much smaller than with state-of-the art protocols.

I. Introduction

Energy consumption in WSN received so much attention that we might expect the problem to be solved. However, when looking at real implementations of WSNs and the actual applications they are being deployed for, it is easy to realize that optimizing energy consumption is still an open problem.

Idle listening, where nodes are waiting for transmissions to happen, is known to be one of the key sources of energy waste. It is clear that some form of coordination that avoids nodes to even *hear* communications they are not interested in can represent a major leap in energy saving. This is achieved by some existing MAC schemes in a rather complex fashion, typically requiring coordination of explicitly signaled communication schedules, making use of RTS/CTS frames, long preambles or time synchronization. We present a method for attaining such *de*-synchronization of listening periods in a simple fashion, without requiring any complex coordination overhead. We deem this concept can be applied to any ad-hoc, on-demand wireless network where nodes are battery powered; however for the sake of simplicity we restrict our analysis here to Wireless Sensor Networks (WSN) scenarios.

When we talk about 'applications', we mean the joint level of application and routing, since in the most efficient WSN systems these two layers are tightly coupled, due to the fact that routing in WSN is data-driven and not address-driven: a packet should be routed to a node based on its content, the route availability, overall network optimization considerations (e.g., avoiding nodes that are short in energy), etc.

The MAC that we propose has a very simple interface with the upper layers and has almost no features that make it specific for an application and as such not general purpose. Still, based on the very reasonable assumption that the application/routing level can often predict the generation time of the next packet with good precision, it turns out to be a sophisticated cross-layer design that allows for a very simple, distributed implementation over the extremely common CSMA-based IEEE 802.15.4 PHY/MAC protocol[1]. The performance gain that can be achieved by waking receivers only when they really need to receive packets is astounding, but easily understood considering that the energy wasted to listen to unwanted packets grows linearly with the number of nodes that are within hearing range.

In the remainder of the paper, we first discuss recently proposed low-energy MAC protocols in Sect. I-A. Then, in Sect. II, we formalize the problem and introduce a notation that is useful to design SESAM, which is described in Sect. III. Sect.IV presents results and simulations, comparing them with a simplified version of a B-MAC-like protocol, and Sect. V ends the paper, discussing the future evolution of this work.

A. Related Work

An extensive amount of work has been done on energy conserving MAC protocols. The common theme in this work is that nodes should be put to sleep as often as possible, and only wake up for receiving packets when it is really necessary to do so.

Existing approaches can be categorized as synchronous and asynchronous, although there are also some hybrids. In what follows, we give some examples. A thorough (albeit somewhat outdated) overview is given in [3].

Synchronous protocols are based on appropriately scheduling transmissions and receiver wake-up periods. Here, the most common scheme is probably S-MAC [11], where all senders and receivers briefly wake up together to contend for the channel, and spend most of the time sleeping instead of listening. An *overhearing avoidance* scheme in S-MAC

[1]The IEEE 802.15.4 specification defines the entire PHY/MAC suite, with a possible interface toward the ZigBee alliance protocol stack; however most of the available hardware for WSN uses the PHY and low-level (framing and CSMA) MAC protocols of 802.11.4 supported by chips like the CC2420 by TI, and then customizes the upper part of the MAC protocol for embedding within the node operating system, e.g., TinyOS or Contiki.

enables nodes go to sleep for the duration of a transmission upon hearing an RTS or CTS packet that is not intended for them. This way, a receiver does not waste a large amount of power by unnecessarily listening to irrelevant frames. S-MAC has been enhanced in T-MAC [10], where the time slots are terminated when the nodes do not have anything to send. Being in the same category as these two schemes, SESAM improves upon them by desynchronizing groups of senders and receivers without keeping track of an explicit schedule or requiring RTS/CTS frames.

Some protocols go a step further than scheduling nodes to act in sync by using TDMA. TRAMA [8], for example, divides time into alternating random access and scheduled access phases. During the random access phase, nodes exchange information about their two-hop neighborhood as well as transmission schedules, which are then used in the scheduled access phase. FLAMA [7] is an improvement (and simplification) of TRAMA. It does not require explicit schedule announcement during the scheduled access periods, and relies on knowledge about application-specific traffic patterns (called "flows") which are exchanged during the random access phase.

In Z-MAC [9], another TDMA based protocol, schedules are initially assigned during a setup phase, and this process is only repeated if a significant change in the network topology occurs. Funneling-MAC [1], which exploits the typical tree-upstream data flow in sensor networks for TDMA schedule distribution, was shown to outperform Z-MAC. Both Funneling-MAC and Z-MAC are hybrids between TDMA and random access in that they fall back to CSMA/CA in certain situations.

Asynchronous energy saving MAC protocols are commonly based on Low Power Listening (LPL). Here, the sender transmits a preamble before sending the actual data, and potential receivers regularly check whether the channel is busy or not. In B-MAC [6], this preamble is longer than the sleep period of receivers, and so a receiver will not miss the preamble when it wakes up. Once it detects a preamble, it stays awake to receive the ensuing data frame. This method has the obvious caveat of spending energy on preamble transmission, and preambles can be very long in case of long sleep schedules.

This issue is addressed in various ways in other work. WiseMAC [4], for example, assumes the existence of an access point. This way, the preamble length can be significantly reduced. Not all methods for shortening the preamble require an access point. Two ideas are combined in X-MAC [2]: firstly, in X-MAC, preambles contain a destination address. This allows nodes which wake up and see the preamble but are not the intended receivers to immediately go back to sleep. Secondly, the preamble contains short gaps in time during which a receiver can send an ACK to the sender. Upon reception of an ACK, the sender immediately transmits the data frame, thereby shortening the preamble.

SCP-MAC [12] could be seen as a hybrid approach, as it uses LPL instead of a synchronized time slot for communication, yet it synchronizes the wake-up periods of receivers in order to reduce the necessary preamble length. Another example of a hybrid mechanism can be found in [5], where the authors combined T-MAC with LPL.

This overview shows that the research community has gone to great lengths in trying to improve the energy efficiency of MAC schemes, adding complexities that seem unnecessary in the light of SESAM, which attains a drastic energy saving by applying a simple distributed scheduling algorithm, without requiring TDMA or LPL.

II. Notation and problem formulation

Let $T_{i,j}(t)$ be the (one hop) traffic relation from node i to node j, and let $\{t'_{i,j}(l); l \in \mathbb{N}\}$ be the point process of the arrival times of packets at the MAC interface of node i with one-hop destination node j. We assume that the application/routing layer is able to predict with good approximation $t'_{i,j}(l+1)$ at time $t'_{i,j}(l)$, i.e., when the application delivers packet l to the MAC, it can also predict the next time when it will have a packet with the same destination. This is trivial for any application with constant sampling time, but it is also true for most systems where the application decides the next sampling before putting the sensor back to sleep.

Let $t_{Tx}(l)$ be the transmission time of packet l and $\mathcal{M}$ be the set of nodes in the entire WSN.

We can model the energy node that i consumes as

$$\mathcal{E}_i(t) = P_I \cdot t + P_T \cdot t_T + P_R \cdot t_R + P_S \cdot t_S \qquad (1)$$

where P_I is the power a node uses during sleep periods and is present at any time t, P_T is the power a node uses during transmission (t_T being the total accumulated transmission time), P_R is the power a node uses while receiving (t_R is the total receiving time), and P_S is the power needed for sensing the channel (t_S is the total time spent sensing). Product data sheets show that $P_I << P_T \simeq P_R \simeq P_S$, so that WSN energy consumption is dominated by the communication functions, which are well captured by the couples (P_T, t_T), (P_R, t_R), and (P_S, t_S); besides P_I is not controllable and is constantly reduced as technology improves, while the power related to communications is intrinsic to the process and improves only marginally with technology.

The goal of an energy-aware MAC layer is to deliver all packets while minimizing energy consumption. Formally this can be described as finding a proper schedule for the process

$$\{t'_{i,j}(l) + t^{st}_{i,j}(l), t^{w}_{i,j}(n), t_{Tx}(l); i, j, l, n \in \mathbb{N}\} \qquad (2)$$

which minimizes the overall energy consumption over a time interval T

$$\int_O^T \left[\sum_{i \in \mathcal{M}} \mathcal{E}_i(t) \right] dt. \qquad (3)$$

In the process above $t'_{i,j}(l)$ and $t_{Tx}(l)$ are intrinsic traffic properties, so the only variable that can be controlled is $t^{st}_{i,j}(l)$, a scheduling/staggering delay that can be used to avoid collisions and separate traffic relations. The problem is complicated by the fact that node j must be listening when node i sends the packet, so that $t'_{i,j}(l) + t^{st}_{i,j}(l)$ must be within a given interval preamble interval τ_{pr} from $t^{w}_{i,j}(n)$ for some

n, where $t^w_{i,j}(n)$ is the n-th wake-up time of node j, possibly specific for the relation $T_{i,j}(t)$.

Our goal is to find simple, local rules that would allow the reduction of energy consumption, while requiring minimal coordination and signaling overhead.

Since the power consumption for transmitting, receiving or sensing the channel is approximately equal (see Table I), the goal is to minimize three factors: i) useless (re)transmissions, ii) receiving packets which are not for the node, and iii) sensing the channel without need. Constraints are: a) no global coordination, but only pairwise (i, j) implicit signaling; b) self-bootstrapping properties for new nodes entering the system and for the activation of a new traffic relation $T_{i,j}(t)$ at time t.

III. Protocol Design

We now proceed to define, step-by-step, a simple protocol based on dynamic, per-event contention resolution that minimizes power consumption and requires neither explicit signaling of traffic patterns nor explicit inter-node coordination to define the access times, i.e., using the notation above, $t^{st}_{i,j}(l)$ is determined by nodes i and j independently from any other node k in the WSN.

A. Basic functions

The system is based on low-level real-time MAC functions able to do CSMA and generate acknowledgments. If Collision Avoidance capabilities (CSMA/CA) are present they can be used to improve some aspects of the protocol, but they are not necessary, and will not be considered here. The additional overhead needed to send coordination information (including the next transmission time) in each data frame is $h_s = 5$ bytes.

B. Elementary coordination for a single relation $T_{i,j}(t)$

Assume that $T_{i,j}(t)$ is active, i.e., i is sending packets to j with regularity so that

$$t'_{i,j}(l+1) - t'_{i,j}(l) < \Delta_{\max}, \forall l. \tag{4}$$

where $\Delta_{\max}$ is a parameter identifying the maximum time period that a traffic relation can be idle and still be considered active. How to deal with the cases when $T_{i,j}(t)$ is not active will be discussed in Sect. III-C.

Since node i knows $t'_{i,j}(l+1)$ when transmitting packet l, all that is needed for coordination is transmitting in the header of packet l the difference $\Delta_t(l+1) = t'_{i,j}(l+1) - t'_{i,j}(l)$ so that node j knows when to wake up to receive the next packet. $\Delta_t(\cdot)$ is expressed in μs.

The actual waking time will be

$$t^w_{i,j}(l+1) = t'_{i,j}(l+1) - \tau_s \tag{5}$$

τ_s being the sensing time, normally a parameter of the hardware or PHY protocol, while the actual time to wake up and then begin transmission for node i is

$$t''_{i,j}(l+1) = t'_{i,j}(l+1) - \tau_{\text{pr}}/2 - \tau_s \tag{6}$$

with τ_{pr} set to account for receiver synchronization plus residual clock drifts.

If transmissions are successful, then the average energy consumption for the transmission of packet l is

$$\mathcal{E}(l) = \mathcal{E}^{Tx}_i(l) + \mathcal{E}^{Rx}_j(l)$$

$$\mathcal{E}^{Tx}_i(l) = \mathcal{E}_S + \mathcal{E}_{R-T} + P_T(\tau_{\text{pr}} + t_{Tx}(l)) + \mathcal{E}_{T-R} + P_R\tau_a$$

$$\mathcal{E}^{Rx}_i(l) = \mathcal{E}_S + P_R(\tau_{\text{pr}}/2 + t_{Tx}(l)) + \mathcal{E}_{R-T} + P_T\tau_a$$

where $\mathcal{E}_S$ is the energy required by the sensing function, including the radio power-up, $\mathcal{E}_{T-R}$ and $\mathcal{E}_{R-T}$ are the (fixed) energies required to switch from transmission to reception and vice-versa, and τ_a is the (fixed) time required to send an acknowledgment. Fig.1 exemplifies the transmission timing (and consequently the energy consumption) for the transmission of a single frame.

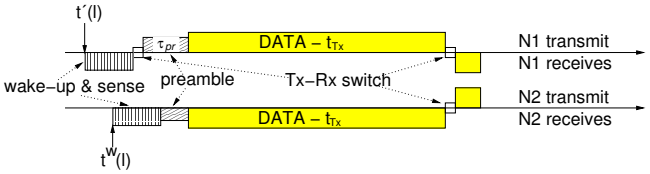

Fig. 1. Timing of a single successful packet transmission

So far, we described the behavior when no channel contention occurs, nor there are errors or collisions. The key point of coordination is dealing with these cases. Since it is not possible to distinguish a transmission error from a collision, and since a collision and a channel contention both mean that two different active traffic relations $T_{i,j}(t)$ and $T_{h,k}(t)$ are for some reason synchronized, the node behavior is the same in all cases. Note that nodes i, j are coordinated because $T_{i,j}(t)$ is active, but have no information whatsoever about $T_{h,k}(t)$[2].

Node i and j simply behave as a 0-persistent CSMA, rescheduling their transmission in a later time. Contrary to 0-persistent CSMA, however, the goal of the transmission rescheduling is not to de-correlate the access in time, since the assumption is that the channel is not overloaded, but to de-correlate with another traffic relation that may have, whatever the reason, reached a contention or synchronization. The random retransmission delay should therefore be fairly short, and in any case the retransmission should occur before the next packet of $T_{i,j}(t)$ is scheduled for transmission.

Whenever the transmission of l cannot occur because the channel is already occupied or because it is not successful for any other reason (including collisions) both i and j recompute their intended transmission time as follows

$$\begin{aligned} t'_{i,j}(l) &= t'_{i,j}(l) + t^{st}_{i,j}(l) \\ t^{st}_{i,j}(l) &= D(\Delta_t(l)) \\ \Delta_t(l) &= \Delta_t(l) - D(\Delta_t(l)) \end{aligned} \tag{7}$$

[2]For the sake of simplicity we describe the case when two traffic relation have a conflict, but cases with more contenders are just the same — they simply have a larger probability of non successful resolution.

and immediately go back to sleep. $D(\Delta_t(l))$ is a sample from a suitable pseudo-random sequence which is known to both i and j, e.g., a sample of a PN (pseudo-noise) generator initialized with the same seed (see Sect. III-C) properly normalized to fall in the interval $[t_{Tx}^{\max}, t'_{i,j}(l+1)]$, where $t_{Tx}^{\max}$ is the maximum transmission time for any packet in the system. The actual distribution of $D(\Delta_t(l))$ can be tuned to best fit the system, but typically it will be a truncated exponential or a triangular distribution favoring shorter delays.

If for any reason the access fails again, then the procedure (7) is repeated again until $\Delta_t(l) \leq 0$, in which case the packet l is sent once more instead of packet $(l+1)$ and, if it fails once more, the traffic relation $T_{i,j}(t)$ is aborted, packet l is discarded and packet $(l+1)$ is transmitted as the first of a newly activated relation.

If at any access attempt the transmission is successful, then node i will stabilize and maintain the scheduling delay thus computed for all subsequent packets of the relation $T_{i,j}(t)$. Other options, like trying to randomly sample the space between the old and the new schedule to recover some of the delay may be explored.

a) Missing acknowledgments: The procedure described above works smoothly as long as the transmitter and the receiver have the same 'view' of the communication. Unfortunately there is one case when this is not true: the acknowledgment is corrupted. In this case the transmitter i will proceed with procedure (7), while the receiver j, having correctly received packet l will not, mis-aligning the pseudo-random sequences.

Unfortunately, at this point i will keep trying to send packet l following the procedure described above, and j will not wake up to listen, until at node i $\Delta_t(l) \leq 0$. In the meantime j will schedule its next waking time at the original $t'_{i,j}(l+1)$.

A simple trick allows recovering the situation without resetting $T_{i,j}(t)$. When $\Delta_t(l) \leq 0$, node i will still send the packet l at the time $t'_{i,j}(l+1)$ originally planned for packet $(l+1)$. If this transmission is successful, then it also contains the new intended $t'_{i,j}(l+1)$, so j can re-align in time and needs to do nothing more apart from signaling in the ACK the fact that the packet is duplicated; node i instead re-aligns its pseudo-random generator to the value it had before the corrupted acknowledgment spawned the misalignment.

C. Housekeeping and bootstrapping

The procedure described in Sect. III-B is the fundamental algorithm devised to reduce energy consumption. However, for a MAC to work in a *real* system, it also needs to address issues like allowing new nodes to join the network, provisioning for broadcast communications and transmission of unforeseen messages, as well as steady flow setup.

Maintenance of a coordinated network requires some broadcasting possibilities, i.e., all nodes within transmission range listening at the same time. Indeed we 'destroyed' the intrinsic broadcasting capabilities of CDMA in order to reduce energy consumption. To restore it, it is enough to introduce a common listening time, which can be used to setup traffic relations

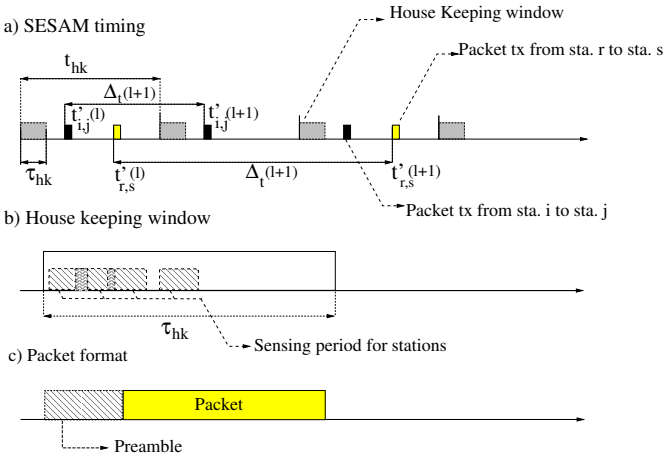

Fig. 2. Global time framing of SESAM protocol

$T_{i,j}(t)$, to send broadcast/flooding messages (e.g., routing and topology management), and to send alarms, since most applications will have sampling rates in the order of tens of seconds if not minutes, but we want a system that is more reactive to alarms and changing conditions. We call these broadcast periods *housekeeping* (hk) periods.

Fig. 2 presents the global time-framing of SESAM. Notice that, contrary to most systems and intuition, the HK repetition time t_{hk} is in general smaller than the typical interval between application packets. t_{hk} depends on the environment (for instance on how dynamic the routing is), the technology (how stable are the clocks) and the application scenario (e.g., how 'fast' an alarm must be). We think that in general this latter point dominates, requiring t_{hk} between 1 and 10 s.

One further point is how long the listening (sensing) period τ_{shk} of nodes should be at housekeeping. The goal is to ensure that all nodes will be able to receive a broadcast packet, and its duration is related to the drift of clocks at nodes. Standard clocks have a stability which is typically better than 10^{-7}, so assuming a safe 10^{-6} stability implies that some form of synchronization must be enforced within $T_{sy} = \frac{1}{2}10^6\tau_{shk}$. Adopting $\tau_{shk} = 1$ ms requires that a synchronization frame must be sent every few hundred seconds; node timers are reset at the end of the synchronization frame transmission. If a routing protocol is present, we can safely assume that the heart-beat of the protocol will be smaller than a few hundreds seconds and it can provide the synchronization. Indeed, any transmission in a housekeeping period can be used for synchronization. If no transmissions are present, nodes can simply generate synchronization frames at housekeeping intervals with probability $P_{sy} = \frac{2t}{T_{sy}}$ starting $\frac{1}{2}T_{sy}$ s after a successful transmission in a housekeeping period. This will provide the required synchronization with a negligible collision probability and deterministic transmission of the synchronization frame within T_{sy}.

Broadcast frames transmitted in housekeeping intervals should contain information useful for network management and synchronization, including at least τ_{shk}, t_{hk}, T_{sy}, and

an identifier of the collision domain to allow some form of coordination among contiguous domains, e.g., allowing a node within multiple different collision domains to distinguish between them and properly coordinate access in all of them.

Multiple collision domain operation is very interesting and poses additional challenges, which unfortunately cannot be discussed here for lack of space, but we report some sample results in Sect IV-D. If multiple frames must be sent during the same HK interval, they can normally be sent one after the other in sequence. If no collision avoidance means are present, this may end up in fairly high collision probabilities. If a collision occurs, packets must be rescheduled in subsequent HK intervals with a proper backoff mechanism.

Summarizing, housekeeping periods are used to:

1) Let nodes join the network. A node has to continuously listen to the channel until it overhears a synchronization frame, then it can immediately transmit its own request to join the network;
2) Allow broadcast, routing, and other generic information transmission;
3) Enable the setup of application related traffic flows $T_{i,j}(t)$ between nodes.

Traffic related to flows $T_{i,j}(t)$ should avoid scheduling transmissions at housekeeping times to avoid channel contention (not necessarily collisions). This is safely obtained by avoiding the scheduling of flow packets for a few (say 3–4) packet transmission times after the starting of housekeeping periods. We call this time τ_{hk}.

IV. Performance Analysis

Before proceeding to implement our protocol within TinyOS in a real WSN, we want to have an initial estimate of its efficiency, possibly compared with some alternative approach. In order to do this, we first make some rule of thumb computations for very simple reference scenarios, then we run some simulations with more realistic assumptions. We consider the 3Mate devices, which are derived from the Berkeley Tmote design and produced by TRETEC S.r.L (www.3tec.it), a local producer. Table I reports the fundamental parameters of these devices. The numbers are rounded for the sake of simplicity, we assume -5 dBm transmission power which corresponds to 45mW, and we set $\mathcal{E}_{T-R} = \mathcal{E}_{R-T} = 0$ because these are identical for all protocols and only represent a negligible constant factor. Instead, we consider the energy to power-up the radio from the idle conditions, which is indeed the dominating factor in the sensing function[3].

We compare SESAM with two versions of a B-MAC–like[4] protocol. For all protocols we consider acknowledged transmission and absence of collision avoidance procedures. Moreover, in simulations, we assume absence of channel errors and we consider 'lost' packets that either collide or are not transmitted before the next transmission request arrives to the MAC from upper layers.

PHY – low level MAC	802.15.4
P_I – idle power	1 μW
P_S – sense power	30 mW
P_R – Rx power	60 mW
P_T – Tx power	25 to 50 mW
τ_s	5 ms

TABLE I
3MATE DEVICE CHARACTERISTICS

A. Benchmark MAC protocols

B-MAC is an elementary protocol based on the idea of periodically sensing the channel to spare sensing energy and transmitting a long preamble to 'capture' sleeping receivers when they wake up. B-MAC was shown in [6] to have a very good energy efficiency compared to other protocols like S-MAC, thus a comparison with models of this protocol is of the utmost interest.

b) BenchMAC-0: Upon plain CDMA we insert a low power listening (LPL) functionality which enables nodes to sleep most of the time, and wake up periodically to sample the channel status. The sampling time is τ_s since it depends on the hardware and there is no reason to artificially increase it. The sensing repetition period is t_{lp} and can be set to tune the protocol: lower values of t_{lp} increase channel capacity and energy consumption, and higher values do the opposite. Nodes are not coordinated and hence not in phase.

A node wishing to transmit senses the channel, then if it is free starts transmitting a preamble of duration $(t_{lp} + 2\tau_s)$, which ensures that all nodes will wake up in time to listen to the following packet. The recipient node will receive the entire packet, while all other nodes will turn off the radio immediately after receiving the header, which tells them that the packet is intended for another node.

If a node wishing to transmit senses the channel busy, it simply re-schedules the transmission after exactly t_{lp}. If a packet is not transmitted before a new one is offered to the MAC by higher layers it is discarded.

This protocol is collision free, in the sense that collisions can occur only if two scheduled transmissions happen within τ_s of each other. Since τ_s is very short compared to the average inter-packet time the collision probability is very small.

c) BenchMAC-1: This is the 1-persistent version of the protocol. The differences with BenchMAC-0 are: i) if the channel is sensed busy (also by a preamble), then the node

[3] P_I and τ_s in Table I are somewhat delicate parameters because they include several different operational details of the hardware. For instance, the actual sampling time of the channel for the CC2420 radio interface is only 128 μs, but this is only true if the radio is already on, and powering up the radio requires from 300 to 600 μs, but with a different and variable power level. Thus both P_I and τ_s represent average "reasonable" values that will need verification on actual protocol implementations.

[4] We do not claim that the simplified MACs we describe and use here are *exactly* representative of B-MAC, a sophisticated and already implemented protocol; we are just trying to gain insight into possible solutions and tradeoffs for energy consumption.

Channel rate	250 kbit/s
Frame header (BenchMAC)	11 bytes
Frame header (SESAM)	16 bytes
ACK size	11 bytes
Payload size	0–114 (85)
t_{lp}	0.1–10 s (0.5)
t_{hk}	1–10 s (2)
τpr	0.5 ms
τ_{hk}	10-100 ms (20)
Per node av. energy	20–30 kJ (25)

TABLE II
DEFAULT PARAMETERS OF THE CONSIDERED PROTOCOLS AND SCENARIO

waits until the transmission ends and immediately transmits the packet, and as a consequence ii) all the other nodes (including the one that was occupying the channel) must keep sensing the channel after the end of a packet transmission for $2\tau_s$ in order to detect possible additional packets.

This protocol is not collision free, but we can still expect collision probabilities to be low. If collisions occur, the packet can be rescheduled with a random delay within the next t_{lp}.

B. Expected gain

Table II reports the fundamental parameters of SESAM and the benchmark protocols we are considering, including the ranges of parameters that are dependent on the scenario[5]; default parameters used in simulation and evaluation are reported in parentheses. Other parameters are consistent with IEEE 802.15.4.

We can compute the energy consumption directly from (1) and (3) (and obviously the protocol dynamics) in the hypothesis that collision probabilities are negligible.

First of all we consider the plain CSMA option. In this case the energy is dominated by sensing, which is always on, with just a 'small' additional consumption to transmit and receive data. Starting from data in Table I, assuming negligible traffic, it is easy to compute that the energy consumption per node per day is around 2.5–3.0 kJ/day: the lifetime of nodes, with energy availability as in Table II, will be little more than one week, even without traffic!

Next we consider SESAM and the benchmark protocols sketched in Sect. IV-A. In the approximation of negligible contentions and collisions, the persistency of the protocol does not influence energy consumption, and we can consider a single "BenchMAC" for comparison. The protocol parameters are those listed in Table II, and we examine the energy consumption per node per day as a function of the number of nodes within hearing distance, and of the traffic load in packets per minute per station. In this theoretical analysis we only consider application level traffic, so that the actual consumption including routing, broadcast and maintenance traffic will be higher, as discussed with simulations in Sect. IV-C.

[5]We consider 2 AA batteries. Depending on 'quality' these have between 2000 and 3000 mAh of capacity; taking an average of 2500 yields about 25 kJ of available energy.

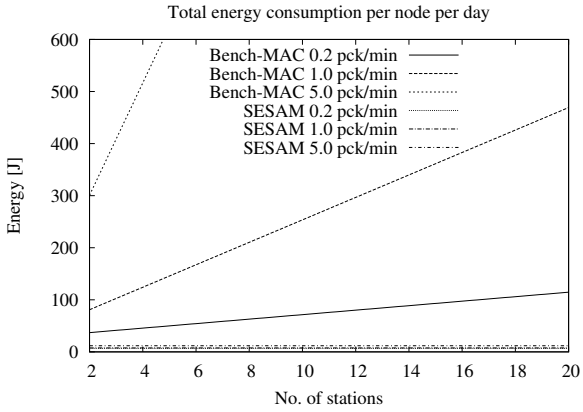

Fig. 3. Energy consumption per day estimated for SESAM and for the benchmark protocol as a function of the number of nodes within hearing range and the traffic load

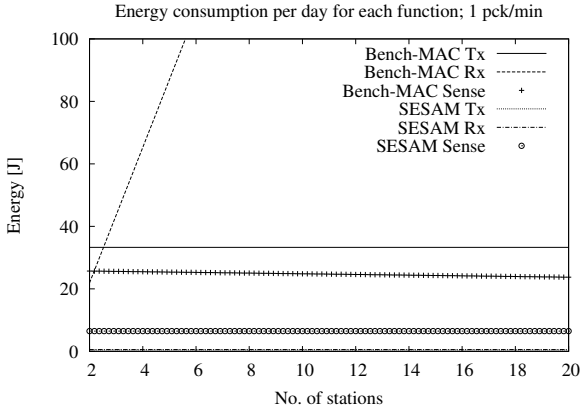

Fig. 4. Split of the energy consumption for the three principal functions: transmitting, receiving and sensing the channel

Fig. 3 reports the energy consumption. As expected the consumption grows with the increase of traffic for both protocols. The huge difference is instead that the energy consumed by SESAM is not only much smaller thanks to the coordination that avoids useless sampling of the channel, but it is also independent of the dimension of the neighborhood, while BenchMAC energy consumption grows linearly with it. Given the characteristics of a CSMA protocol without coordination this is not a great surprise, and the explanation comes from separately considering the energy consumed by the three functions: transmitting, sensing and receiving.

This analysis is done in Fig. 4 for the load of 1 pck/min. Transmitting (and receiving) a packet of 85 bytes requires around 4 ms, including all headers and preambles. Thus the *actual* transmitting (and receiving) time in one day with 1 pck/min is less than 6 s, requiring less than 0.5 J per each function. This is the reason why for SESAM Tx and Rx functions in Fig. 4 are almost indistinguishable from the x axis. BenchMAC waste a lot of energy in preamble transmission, but this cannot be avoided because sleeping times are not synchronized.

Indeed, the key difference is receiving power: SESAM coor-

dination avoids the waste of receiving useless preambles and headers just to realize that the packet is not intended for the node. This behavior is the reason why Bench-MAC receiving energy grows with the neighborhood, and, with preambles much longer than transmission times it is indeed dominating. Notice that reducing preambles is not easy: reducing t_{lp} increases the energy wasted in sensing. The difference in sensing energy is due to the fact that idle sensing for SESAM is limited to housekeeping, while in BenchMAC it also supports application traffic. Finally, BenchMAC sensing energy reduces slightly with the neighborhood size because sensing time is reduced by the increase time occupancy of the channel.

The only way a B-MAC like protocol can be really energy efficient is with a hardware that significantly reduces the energy required to power-up the radio front-end, which dominated τ_s: keeping the ratio t_{lp}/τ_s constant, the wasted energy decreases roughly linearly with τ_s, but this seems hardly a goal that a protocol can achieve, rather it is a technological improvement that all protocols will exploit.

C. Simulations

The simple analysis in Sect. IV-B is useful for understanding the basic reasons why SESAM is really savvy in using energy, however some more realistic evaluation is needed.

We have developed an event driven simulator based on PeerSim[6] that evaluates the time nodes spend in different states, based on the data sheets of the 3Mate devices and the protocols FSM. Energy consumption follows trivially from the time.

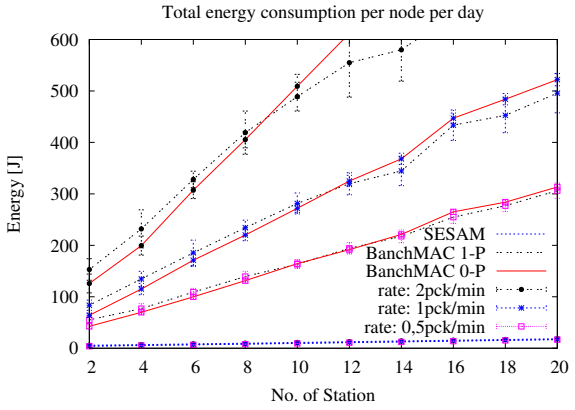

Fig. 5. Energy consumption per day measured via simulation estimated for SESAM and for BenchMAC-0 and BenchMAC-1 as a function of the number of nodes within hearing range and the traffic load

Fig. 5 reports the energy consumption per node per day as a function of the neighborhood size and per node traffic load. Applications generate traffic at constant periods, but nodes are not synchronous with one another, and channel access is managed only through SESAM distributed coordination. All simulations are run 10 times, lines report the average value while error bars refer to the minimum and the maximum measured. This figure should be compared with Fig. 3, but here we also have broadcast traffic, which explains why also SESAM consumption increases with the number of stations. The broadcast traffic is set to 0.1 pck/node/minute for the benchmark protocols, assuming that this traffic is due only to keepalive and routing management protocols. In SESAM instead we set the broadcast traffic to 1 pck/node/minute, i.e., 10 times higher, to account for opening and closing of traffic relations and synchronization on top of keepalive and routing.

The energy-saving properties of SESAM are fully confirmed and indeed the lifetime of the network is greatly extended and turns out to be almost independent of the the number of stations within hearing range. BenchMAC-0 and BenchMAC-1 are equivalent. Just to give an idea, assuming a traffic of 1 pck/min and a neighborhood of 8 nodes, under the same conditions for which plain CSMA has a lifetime of one week, BenchMAC-0,1 would have a lifetime of about 3 months and SESAM of close to 10 years, avoiding that the communication function is the energy bottleneck of the system.

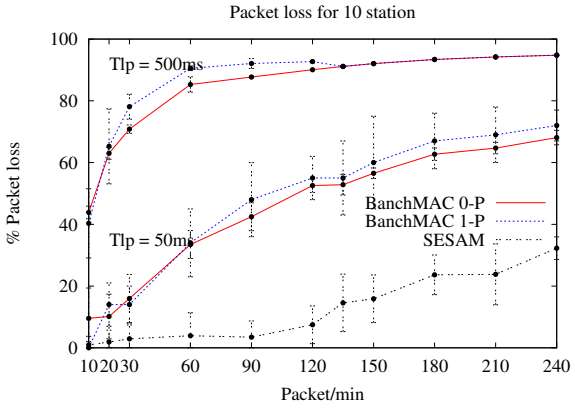

Fig. 6. Packet loss rate for a 10-node network as a function of the load per node for SESAM, BenchMAC-0, and BenchMAC-1

An energy saving protocol is useful only if it also guarantees good communication properties. Given the design of the protocol, we can easily infer that its performance is generally better than a B-MAC-like protocol. Fig. 6 reports the loss rate for a 10-node WSN as a function of the load per node in packet/min. The loss rate is computed over 10-day runs, which are clearly unreasonable for such high loads, but are useful to understand the behavior of the protocols under stress. Recall that packets are considered lost upon collision or if they cannot be transmitted before the next packet is offered to the MAC by the application. Due to the long preamble that must be transmitted both BenchMAC-0 and BenchMAC-1 saturate the channel for medium loads, leading to very high loss rates. SESAM instead bears loads around 20 times larger than BenchMAC with $t_{lp} = 500$ ms and 10 times larger than BenchMAC with $t_{lp} = 50$ ms.

[6]PeerSim (http://peersim.sourceforge.net/) is a simulator that was originally conceived for P2P systems, however, its event-driven engine and easy to modify classes make it suitable for the simulation of any system of equal objects, exactly like a WSN.

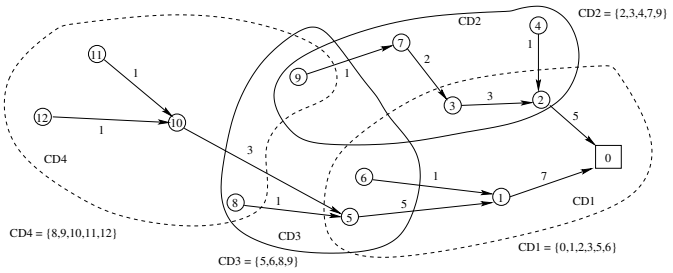

Fig. 7. Sample WSN setup with 13 nodes defining four collision domain and applcation level routing that delivers traffic to a sink; numbers on directed connections are the loads in pck/min

	SESAM	BenchMAC-0	BenchMAC-1
energy [J]	4.2	120.1	131.8
st. dev. [J]	0.0	2.6	3.4
loss rate [%]	0	3.3	2.3

TABLE III
PER DAY ENERGY CONSUMPTION AND LOSS RATE OF THE THREE PROTOCOLS IN THE SCENARIO DEPICTED IN FIG. 7

D. Multiple Collision Domains

The most critical working conditions for a CSMA-based WSN are with a wide area coverage using the same frequency channel. In these conditions there are multiple sensing/collision domains that challenge the simple statistical coordination of CSMA: hidden terminals are the rule, not the exception! In order to test SESAM in these more challenging situations, we set up a multi collision domain topology, shown in Fig. 7, which corresponds to a possible sensing scenario where all sensors (twelve in our example) report regular data to a sink (the square node 0). Fig. 7 shows the logical (directed) tree resulting from the application level routing strategy, together with the four collision domains (CD1–4) that result from the propagation conditions of the environment.

Table III reports the energy consumption and loss rate per day of sensing of the three protocols in the scenario depicted in Fig. 7. Note that also in this case SESAM has the best performance.

V. CONCLUSIONS AND FUTURE WORK

In this work we have presented a novel approach for energy-saving MAC protocols for WSN, looking for distributed, loose coordination of nodes trying and to avoid: i) the transmission of long preambles; ii) the reception of data/headers/preambles that are useless because the destination is not the present node; iii) the complexity of coordinated synchronization typical of TDM protocols.

The result is SESAM a new protocol which has the potential of making communications in WSN a non-problem as far as energy is concerned. Simple analysis and initial simulation results indicate that the energy consumption per node per day can be as low as 5–10 J for typical communication patterns and network sizes, leading to years of lifetime out of standard AA or AAA batteries.

Indeed, more interesting than long lifetime out of batteries, is the fact that SESAM enables cheap energy scavenging systems with accumulation in off-the-shelf ultra-capacitors. For instance, a 10 F, 3 V ultra-capacitor yields 90 J of energy, enough for multiple days.

Future work aims at actual implementation of SESAM as well as more detailed simulations to fully understand details before this step. The first implementation platform is TinyOS on a 3Mate and Tmote hardware. After implementation, a comparison with existing protocols will be carried out.

ACKNOWLEDGMENTS

The authors wish to thank Alessandro Russo for providing the initial framework of the simulator as well as precious advice on how to manipulate it, and bend it to the needs of SESAM. The mobility between Innsbruck and Trento is provided by the Bozen-Innsbruck-Trento (BIT) School and funded by local governments. Matteo Nardelli is supported by the TRITon project (http://triton.disi.unitn.it).

REFERENCES

[1] Gahng-Seop Ahn, Se Gi Hong, Emiliano Miluzzo, Andrew T. Campbell, and Francesca Cuomo. Funneling-mac: a localized, sink-oriented mac for boosting fidelity in sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 293–306, New York, NY, USA, 2006. ACM.

[2] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, New York, NY, USA, 2006. ACM.

[3] Ilker Demirkol, Cem Ersoy, and Fatih Alagoz. Mac protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121, April 2006.

[4] Amre El-Hoiydi and Jean-Dominique Decotignie. Low power downlink mac protocols for infrastructure wireless sensor networks. *Mob. Netw. Appl.*, 10(5):675–690, 2005.

[5] G. P. Halkes, T. van Dam, and K. G. Langendoen. Comparing energy-saving mac protocols for wireless sensor networks. *Mob. Netw. Appl.*, 10(5):783–791, 2005.

[6] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM.

[7] V. Rajendran, J.J. Garcia-Luna-Aceves, and K. Obraczka. Energy-efficient, application-aware medium access for sensor networks. In *Proceedings of IEEE MASS 2005: 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, Washington; D.C.; USA, November 2005.

[8] V. Rajendran, J. J. Garcia-Luna-Aveces, and K. Obraczka. Energy-efficient, application-aware medium access for sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, Washington D.C., November 2005.

[9] Injong Rhee, Ajit Warrier, Mahesh Aia, and Jeongki Min. Z-mac: a hybrid mac for wireless sensor networks. Technical report, Department of Computer Science, North Carolina State University, April 2005.

[10] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, New York, NY, USA, 2003. ACM.

[11] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.

[12] Wei Ye, Fabio Silva, and John Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 321–334, New York, NY, USA, 2006. ACM.