Asynchronous Training in SANET

F. Barsi, F. Betti Sorbelli, R. Ciotti, M.C. Pinotti
Department of Computer Science and Mathematics
University of Perugia
06123 Perugia, Italy
{barsi,pinotti}@unipg.it

A.A. Bertossi
Dept. of Comp. Scie.
University of Bologna
40127 Bologna, Italy
bertossi@cs.unibo.it

S. Olariu
Dept. of Comp. Scie.
Old Dominion University
Norfolk, VA, 23529, USA
olariu@cs.odu.edu

ABSTRACT

Scalable energy-efficient training protocols are proposed for networks consisting of Sensors and Actors (SANET), where the sensors are initially anonymous and unaware of their location. The protocols are based on an intuitive coordinate system imposed onto the deployment area which partitions subsets of the sensor population into clusters. The protocols are asynchronous, in the sense that the sensors wake up for the first time at random, then alternate between sleep and awake periods both of fixed length, and no explicit synchronization is performed between them and the actor. Theoretical properties are stated under which the training of all the sensors is possible. Moreover, an experimental evaluation of the performance is presented, showing that the protocols are lightweight and flexible.

Categories and Subject Descriptors

C.2.3 [**Computer Communication Networks**]: Network Operations—*Network Management*; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Distributed Networks, Wireless Communication*

General Terms

Design Management Algorithms

Keywords

Sensor Network, Training, Actors, Energy Awareness

1. INTRODUCTION

Technological breakthroughs in ultra-high integration and low-power electronics have enabled the development of miniaturized battery-operated sensor nodes (*sensors*, for short) that integrate signal processing and wireless communications capabilities [2]. Together with innovative and focused network design techniques that will make possible massive deployment [15] and sustained low power operation, the small size and cost of individual sensors are a key enabling factor for a large number of applications. Indeed, aggregating sensors into sophisticated computational and communication infrastructures, called *wireless sensor networks*, has a significant impact on a wide array of applications.

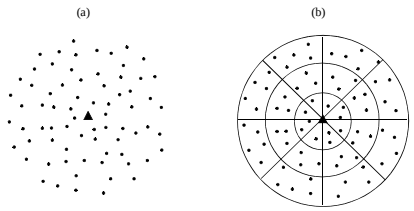

Figure 1: (a) ***Illustrating an actor and surrounding sensors.*** **(b)** ***The trained sensor network.***

Recently, it has been recognized that it would be beneficial to augment sensor networks by more powerful entities. This leads to a heterogeneous deployment including, alongside with the tiny sensors, some entities referred to as *actors* [1] or Aggregation and Forwarding Nodes (AFN) [10]. While the sensors are tasked mainly to sense their immediate neighbourhood, the actors collect, aggregate and fuse the data harvested by the sensors in order to act on the environment in a meaningful way. The resulting augmented version of sensor networks is commonly referred to as Sensor Actor Networks (SANET). As pointed out by Olariu *et al.* [10], the typical mode of operation of an actor is to task the sensors in a disk of radius ρ centered at itself to produce data relevant to the mission at hand. Once this data has been aggregated, the actor has a good idea of what action to take. For instance, Figure 1(a) illustrates a disk around an actor. In this scenario, the actor is equipped with a special long-range radio and has a full range of computational capabilities, can send long-range directional broadcasts to the sensors at distance at most ρ, can receive messages from nearby sensors, and has a steady power supply.

The random deployment results in sensors initially unaware of their location. Further, due to limitations in form factor, cost per unit and energy budget, individual sensors are

not expected to be GPS-enabled. Moreover, many probable application environments limit satellite access. Therefore, individual sensors have to determine their exact geographic location, if required by the application, or else a coarse-grain approximation thereof. The former task is referred to as *localization* and has been extensively studied in the literature [7]. The latter task, referred to as *training*, has been considered in several recent papers by Olariu *et al.* [3, 4, 11, 14, 16]. In particular, they devised some training protocols for sensor networks, which differ on whether or not sensors need some kind of explicit synchronization with the actor. Such training protocols have different performance, measured in terms of total time for training, overall sensor awake time, and number of sensor sleep/wake transitions. In particular, the model in [3, 16] assumes that the (single) actor and the sensors are asynchronous, in the sense that the sensors wake up for the first time at random and then alternate between sleep and awake periods both of fixed length, while no explicit synchronization is performed between them and the actor.

The main contribution of this paper is to further study the task of training, assuming the same asynchronous model as that defined in [3, 16]. The present paper improves on the work of [3], where three *flat* protocols based on linear signal strength decrease were proposed. In the present paper, new protocols are exhibited which rely on a *two-level* approach, where jumps are initially made in the flat protocols and are filled later on. The two-level protocols outperform the flat ones in terms of both the number of sleep/wake transitions and the overall sensor awake time for training.

The remainder of this paper is organized as follows. Section 2 discusses the SANET model and introduces the task of training. Training imposes a coordinate system which divides the sensor network area into equiangular wedges and concentric coronas centered at the actor, as first suggested in [14]. Section 3 reviews the three flat training protocols, called Flat–, Flat, and Flat+, previously proposed in [3], along with their worst-case performance analysis. Section 4 presents the novel two-level approach, which leads to three new protocols, called TwoLevel–, TwoLevel, and TwoLevel+, respectively. Section 5 contains an experimental evaluation of their performance, tested on randomly generated instances, showing that the two-level protocols outperform the flat ones. Finally, Section 6 offers concluding remarks.

2. THE NETWORK MODEL

In this work a SANET is assumed that consists of a single fixed actor and a set of sensors randomly deployed in its broadcast range as illustrated in Figure 1(a). For simplicity, the actor is centrally placed, although this is not really necessary.

A sensor is a device that possesses three basic capabilities: sensory, computation, and wireless communication, and operates subject to the following fundamental constraints:

a. Sensors are *anonymous* – they do not have individually unique IDs;

b. Each sensor has a modest non-renewable energy budget and a transmission range of r;

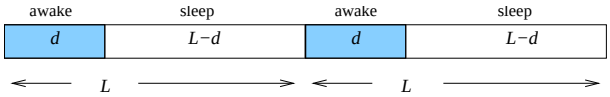

Figure 2: ***The sensor sleep-awake cycle.***

c. In order to save energy, each sensor alternates between *sleep* periods and *awake* periods, as depicted in Figure 2 – the sensor sleep-awake cycle is of total length L out of which the sensor is in sleep mode for $L - d$ time and in awake mode for d time;

d. Each sensor is asynchronous – it wakes up for the first time according to its internal clock and is not engaging in an explicit synchronization protocol with either the actor or the other sensors;

e. Each sensor has no global information about the network topology, but can hear transmissions from the actor;

f. Individual sensors must work *unattended* – once deployed it is either infeasible or impractical to devote attention to individual sensors.

The task of training is essential in several applications. One example is clustering where the set of sensors deployed in an area is partitioned into clusters [2, 5, 12]. As a result of training, we impose a coordinate system onto the sensor network in such a way that each sensor belongs to exactly one cluster. The coordinate system involves establishing [14]:

1. *Coronas*: The deployment area is covered by k coronas $C_0, C_1, \ldots, C_{k-1}$ determined by k concentric circles, centered at the actor, whose radii are $0 < r_0 < r_1 < \cdots < r_{k-1} = \rho$;

2. *Wedges*: The deployment area is ruled into a number of equiangular wedges, centered at the actor, which are established by directional transmission [11].

For the sake of simplicity, in this paper, it is assumed that the corona width is equal to the sensor transmission range r, and hence the (outer) radius r_i of corona C_i is equal to $(i + 1)r$. Moreover, the length L of the sensor sleep-awake cycle is assumed to be no smaller than the number k of coronas. As illustrated in Figure 1(b), at the end of the training period each sensor has acquired two coordinates: the identity of the corona in which it lies, as well as the identity of the wedge to which it belongs. In particular, a cluster is the locus of all nodes having the same coordinates in the coordinate systems [11].

3. THE FLAT PROTOCOLS

This section recalls the basics of the flat corona training protocols presented in [3]. Their goal is that each individual sensor will learn the identity of the corona to which it belongs, regardless of the moment when it wakes up for the first time. To see how this is done, it is useful to assume the time ruled into slots. The sensors and the actor use equally

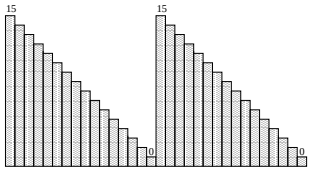

Figure 3: ***The actor transmission cycle.***

```
Procedure Actor (k, τ1);
    for τ := 0 to τ1 − 1 do
        transmit the beacon |k − 1 − τ|k up to
        corona C|k−1−τ|k;
```

Figure 4: ***The protocol for the actor.***

long, in phase slots, but they do not necessarily start counting the time from the same slot.

The idea of the protocol is illustrated in Figure 3. Immediately after deployment the actor cyclically repeats a transmission cycle which involves k broadcasts at successively lower power levels. Each broadcast lasts for a slot and transmits a beacon equal to the identity of the outmost corona reached. Precisely, at time slot τ, with $\tau \geq 0$, the actor transmits the beacon $k-1-|\tau|_k$ with a power level that can reach all the sensors up to corona $C_{k-1-|\tau|_k}$, where $|a|_b$ stands for the non negative remainder of the integer division between a and b (i.e. $|a|_b$ is the same as a modulo b). The actor transmission cycle is repeated for a time τ_1 sufficient to accomplish the entire corona training protocol. The protocol for the actor is shown in Figure 4.

In order to describe the protocol for the sensors, it is crucial to point out that each sensor is aware of the actor behaviour and of the total number k of coronas. Immediately after deployment, each sensor wakes up at random within the 0-th and the $(k-1)$-th time slot and starts listening to the actor for d time slots (that is, its awake period). Then, the sensor goes back to sleep for $L-d$ time slots (that is, its sleep period). Such a sleep/wake transition will be repeated until the sensor will learn the identity of the corona to which belongs, that is, until the sensor will be trained. Each sensor, during the training process, uses a k-bit register R to keep track of the beacons, i.e. corona identities, transmitted by the actor while the sensor is awake. As soon as the sensor hears an actor transmission for the first time, it starts to fill the register R and it is able to learn the actor global time t within the current actor transmission cycle, that is $t = |\tau|_k$. From now on, such a time will regularly increase so that the sensor can derive from t the beacon $|k-1-t|_k$ that the actor is transmitting. Then, in each time slot when the sensor is awake, one entry of R can be always set either to 0 or to 1. In fact, if the sensor hears beacon c, then it sets $R_c = 1$, while if the sensor hears nothing, it sets $R_{|k-1-t|_k} = 0$. Note that the awake sensors which belong to corona c, with $c > 0$, are able to receive any transmitted beacon from c up to $k-1$, whereas they cannot hear the beacons from 0 up to $c-1$. Hence, if a sensor sets $R_c = 0$ (resp., $R_c = 1$) then it belongs to a corona whose identity is higher than (resp., smaller than or equal to) c. Note that only the sensors in corona 0 can

```
    Procedure Flat– (k, L, d);
 1      heard := trained := false; ν := 0;
 2      while wakeup and ¬ trained do
 3          ν := ν + 1;
 4          for i := 0 to d − 1 do
 5              if received beacon c then
 6                  if ¬ heard then
 7                      heard := true,
                        t := k − 1 − c;
 8                  Rc := 1;
 9                  if (Rc = 1 and Rc−1 = 0)
                    or c = 0 then
10                      mycorona := c,
                        trained := true;
11                  t := t + 1;
12              else
14                  if heard then
15                      c := k − 1 − |t|k;
16                      Rc := 0;
17                      if Rc+1 = 1 then
18                          mycorona := c,
                            trained := true;
19                      t := t + 1;
20          if heard then
21              alarm-clock := t := t + L − d;
22          else
23              alarm-clock := alarm-clock + L;
24          go to sleep until the alarm-clock rings;
```

Figure 5: ***The Flat– protocol for a sensor.***

hear beacon 0 and thus they are the only ones which can set $R_0 = 1$. From the above discussion, the following *training condition* holds:

LEMMA 1. *A sensor which belongs to corona c, with $c > 0$, is trained as soon as the entries R_c and R_{c-1} of its register R are set to 1 and 0, respectively. A sensor which is in corona 0 is trained as soon as R_0 is set to 1.*

The resulting sensor protocol, called Flat–, is illustrated in Figure 5. Procedure Flat– mimics the behaviour of the sensor from its first wakeup until it is trained, that is when the identity of the corona to which it belongs is stored into *mycorona*. Each sensor counts the number ν of sleep/wake transitions needed to be trained (line 1), it keeps its local time t, which is initialized when the sensor receives a beacon for the first time from the actor (that is, when *heard* is set to `true` in line 7), and it stores in *alarm-clock* the time when the next sleep/wake transition is planned (line 21–23). After any entry of R is filled, the sensor checks the training condition stated in Lemma 1. Observe that lines 12–19 cannot be executed when $c = k-1$, because the beacon $k-1$ reaches the outmost corona C_{k-1}, all awake sensors hear, and thus they execute lines 6–11. In the procedure, each sensor executes $O(1)$ arithmetic/logic operations per time slot.

In [3], conditions on the parameters k, L, and d were investigated which guarantee that all the sensors are trained by the Flat– protocol, independent of their first wakeup time and from the corona c they belong to. Moreover, an analytical evaluation of the Flat– protocol performance was also provided. Let (a, b) denote the *greatest common divisor* between a and b. If $(a, b) = 1$, let $\left|\frac{1}{a}\right|_b$ be the multiplicative

```
6       if ¬ heard then
7           heard := true, t := k − 1 − c;
7.1         for j := i − 1 downto 0 do
                R_{|c+1+j|_k} := 0;
7.2         for j := ν − 1 downto 1 do
7.3             for h := d − 1 downto 0 do
                    R_{|c+j|L|_k+i−h|_k} := 0;
```

Figure 6: ***The extra instructions for the Flat protocol.***

inverse of a modulo b (e.g. see [6]). Moreover, let ν be the number of sleep/wake transitions required by a sensor to be trained in the worst case, that is the number of sleep/wake transitions required to fill the whole register R. Let ω be the overall sensor awake time and τ be the total time for training. Recalling that a sleep-awake period has length L, a sensor is awake for d time slots per sleep-awake period, and wakes up at time $x < k$, one has $\omega = \nu d$ and $\tau = \nu L + k$. The results shown in [3] can be summarized as follows:

THEOREM 1. *Fixed L, d, and k, if $d < (L,k)$ then there are sensors which cannot be trained by the Flat– protocol; otherwise all the sensors are trained, and:*

1. *If $(L,k) \leq d < |L|_k$, then $\nu \leq \frac{k}{(L,k)} + \left|\frac{1}{L'}\right|_{k'}$, where $k' = \frac{k}{(L,k)}$ and $L' = \frac{L}{(L,k)}$;*
2. *If $|L|_k \leq d < k$, then $\nu \leq \left\lfloor \frac{k}{|L|_k} \right\rfloor + 1$;*
3. *If $d = k$, then $\nu \leq 2$.*

Note that, when $d = (L,k)$ or $d = |L|_k$, since ν equals the upper bound stated in Theorem 1, $\tau = \left(\frac{k}{(L,k)} + \left|\frac{1}{L'}\right|_{k'}\right) L + k$ or $\tau = \left(\left\lfloor \frac{k}{|L|_k} \right\rfloor + 1\right) L + k$, respectively. Referring to Figure 4, it should be clear that τ_1 must be an upper bound on the total time for training, which is derived as $\tau_1 = \nu L + k$ by choosing ν according to the upper bounds given by Theorem 1.

The Flat– protocol can be improved in several ways so as to reduce the number ν of sleep/wake transitions. As a first improvement, recall that, as soon as a sensor hears the actor transmission for the first time, it learns from the beacon the actor global time modulo the actor transmission cycle. Therefore, it can immediately retrieve backwards the coronas which it did not hear and which were transmitted by the actor during its previous awake periods, setting to 0 the corresponding entries of R. The resulting improved protocol, which is called *Flat*, is derived from the Flat– protocol by modifying, as shown in Figure 6, the `if` instruction in lines 6-7 of Figure 5. As a drawback, a sensor may now execute as many as $O(\nu d)$ arithmetic/logic operations per time slot. The time required to perform such arithmetic operations, however, should be negligible with respect to the time slot length, which instead depends on the characteristics of the radio broadcast equipment.

The worst case performance for the Flat protocol is summarized below [3]:

THEOREM 2. *Fixed L, d, and k, if $d < (L,k)$ then there are sensors which cannot be trained by the Flat protocol; otherwise all the sensors are trained, and:*

1. *If $(L,k) \leq d < |L|_k$, then $\nu \leq \frac{k}{(L,k)}$;*
2. *If $|L|_k \leq d < k$, then $\nu \leq \left\lceil \frac{k}{|L|_k} \right\rceil$;*
3. *If $d = k$, then $\nu = 1$.*

Note that, when $d = (L,k)$ and $d = |L|_k$, $\tau = \frac{kL}{(L,k)} + k$ and $\tau = \left\lceil \frac{k}{|L|_k} \right\rceil L + k$, respectively, because ν matches the upper bound given in Theorem 2.

A further improvement to the Flat protocol exploits the fact that when a sensor hears a beacon c, it knows that it will also hear all the beacons greater than c, and thus it can immediately set to 1 the entries from R_c up to R_{k-1}. Similarly, when a sensor sets an entry R_c to 0, it knows that it cannot hear any beacon smaller than c, and thus it can immediately set to 0 the entries from R_{c-1} down to R_0, too. In contrast to the previous protocols, the sensor now fills entries of R relative to beacons not yet transmitted during its awake periods. Therefore, it can look ahead to decide whether it is worthy or not to wake up in the next awake period. If the d entries of R that will be transmitted by the actor in the next awake period have already been filled, then the sensor can skip its next awake period, thus saving energy. The sensor repeats the look ahead process above until at least one unfilled entry is detected among the d entries corresponding to a future awake period. The resulting protocol, called Flat+, is illustrated in Figure 7. Procedure Flat+ makes use of two variables, max0 and min1, which record the largest (smallest, resp.) index of R which has been filled to 0 (1, resp.). When a beacon c is heard, the sensor sets to 1 all the entries from R_c to $R_{\min 1}$ (line 11). When an entry R_c has to be set to 0, then all the entries from $R_{\max 0}$ to R_c are set to 0 (line 19). When the sensor hears the actor for the first time, it stores in max0 the largest entry of R which must be 0 due to its previous awake periods (lines 8-9), and thus it sets to 0 the entries from R_0 to $R_{\max 0}$ (line 10). Finally, at the end of the awake period, the sensor performs the above mentioned look ahead process, properly setting the alarm-clock (lines 24–29).

Clearly, the number ν of sleep/wake transitions of Flat+ cannot be larger than that of Flat. Moreover, when $d = |L|_k$ or $d = (L,k)$, one can find bad instances where ν, in the worst case, is the same for both Flat+ and Flat. For example, when $d = |L|_k$, a sensor which uses Flat+, belongs to corona c, and wakes up when the actor transmits $K_x = c-1$ can never take advantage from the look ahead processing. Hence, the results in Theorem 2 hold for both the Flat and Flat+ protocols.

4. THE TWO-LEVEL APPROACH

The protocols reviewed in the previous section can be further improved by following a nesting approach in which the k coronas are viewed as k_1 macrocoronas of k_2 adjacent coronas each. Precisely, each sensor first learns in which

```
    Procedure Flat+ (k, L, d);
1       heard := trained := false; ν := 0;
        max0 := 0; min1 := k − 1;
2       while wakeup and ¬ trained do
3           ν := ν + 1;
4           for i := 0 to d − 1 do
5               if received beacon c then
6                   if ¬ heard then
7                       heard := true, t := k − 1 − c;
8                       max0 := max{max0, |c + i|_k};
9                       for j := ν − 1 downto 0 do
                            max0 := max{max0,
                                |c + j|L|_k + i|_k};
10                      for h := 0 to max0 do
                            R_h := 0;
11                  for h := c to min1 do R_h := 1;
12                  min1 := c;
13                  if (R_c = 1 and R_{c−1} = 0)
                    or c = 0 then
14                      mycorona := c,
                        trained := true;
15                  t := t + 1;
16              else
17                  if heard then
18                      c := k − 1 − |t|_k;
19                      for h := max0 to c do
                            R_h := 0;
20                      max0 := c;
21                      if R_{c+1} = 1 then
22                          mycorona := c,
                            trained := true;
23                      t := t + 1;
24          if heard then
25              filled:=true,
                alarm-clock := t := t + L − d;
26              while filled do
27                  s := |k − 1 − t|_k, z := 0;
28                  while z ≤ d − 1 and R_{|s−z|_k} is
                        filled do z := z + 1;
29                  if R_z is unfilled
                        then filled := false
                        else t := t + L;
                             alarm-clock := t;
30          else
31              alarm-clock := alarm-clock + L;
32          go to sleep until the alarm-clock rings;
```

Figure 7: ***The Flat+ protocol for a sensor.***

macrocorona it belongs and then refines its training by determining the microcorona inside its macrocorona. Once a sensor learns the index of its macrocorona, say m with $0 \leq m \leq k_1 - 1$, as well as that of its microcorona, say μ with $0 \leq \mu \leq k_2 - 1$, it obtains its actual corona identity as $c = k_2 m + \mu$, where of $0 \leq c \leq k_1 k_2 - 1$. For determining both the macrocorona and microcorona identities, any of the flat protocol variants can be used.

The protocol for the actor is shown in Figure 8. The actor works in two levels and counts the total time in τ. In the first level, the actor cyclically repeats the *macrocorona transmission cycle*, that is a cycle of length k_1 using decreasing powers so as to distinguish different consecutive macrocoronas. In fact, at time slot $z = 0$, the actor starts out by transmitting the beacon $k_1 - 1$ at a power sufficient to reach the sensors up to the outmost macrocorona, that is up to corona C_{k-1}. Next, the actor transmits the beacon $k_1 - 2$ at a power that can be received up to the $(k_1 - 2)$-th

```
Procedure Actor (k_1, k_2, τ_1, τ_2);
    τ := 0;
    for z := 0 to τ_1 − 1 do
        transmit the beacon |k_1 − 1 − z|_{k_1}
        up to corona C_{k_2(|k_1−1−z|_{k_1}+1)−1};
        τ := τ + 1;
    for m := 0 to k_1 − 1 do
        for z := 0 to τ_2 − 1 do
            transmit the beacon |k_2 − 1 − z|_{k_2}
            up to corona C_{mk_2+|k_2−1−z|_{k_2}};
            τ := τ + 1;
```

Figure 8: ***The two-level protocol for the actor.***

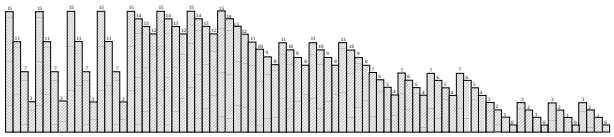

Figure 9: ***The actor transmission cycle for the TwoLevel and TwoLevel+ protocols.***

macrocorona, that is corona C_{k-k_2-1}. For the subsequent $k_1 - 2$ slots, the actor continues to transmit at decreasing powers until it concludes its cycle at time slot $z = k_1 - 1$ with a broadcast that can be received only by the sensors in the 0-th macrocorona, that is, up to corona C_{k_2-1}. The first level lasts for τ_1 time slots, thus repeating $\frac{\tau_1}{k_1}$ times the macrocorona transmission cycle. The time τ_1 is properly chosen to allow all the sensors to be trained with respect to their macrocorona.

In the second level, for each macrocorona, the actor cyclically repeats a *microcorona transmission cycle*, that is one of length k_2 using decreasing powers so as to distinguish different consecutive coronas. Such a microcorona transmission cycle is repeated $\frac{\tau_2}{k_2}$ times so as to allow all the sensors in each macrocorona m to be also trained with respect to their microcorona. Overall the second level of the protocol lasts $k_1 \tau_2$ time slots.

As regard to the protocol for the sensors, it is assumed that each sensor is aware of the two-level actor behaviour and thus of the numbers k_1 and k_2 of macrocoronas and microcoronas, respectively. Each sensor wakes up at time x, with $0 \leq x \leq \min\{k_1, k_2\}$, and repeats its sleep-awake cycle of length L such that $L \geq d \geq \max\{(L, k_1), (L, k_2)\}$. Each sensor uses a k_1-bit register P and a k_2-bit register Q to keep track of the macrocorona and microcorona identities, respectively. As soon as the sensor wakes up at time x, it performs one of the protocol variants, i.e. Flat–, Flat, and Flat+, using its register P to learn its macrocorona identity m. When it has been trained on its macrocorona, it sets its alarm clock to $\tau_1 + (k_1 - 1 - m)\tau_2 + x$ to be ready for the training on its microcorona, and goes to sleep. Reawakened, the sensor performs again the same protocol variant, but now filling its register Q to learn its microcorona identity μ. Clearly, as soon as it knows both m and μ, it derives its corona identity $c = k_2 m + \mu$, and thus it is trained.

Depending on which protocol, Flat–, Flat, and Flat+, is used to train the sensors on each macrocorona and micro-

corona level, three two-level protocols are achieved, denoted by TwoLevel–, TwoLevel, and TwoLevel+. In Figure 9, the macrocorona and microcorona actor transmission cycles are depicted for the TwoLevel and TwoLevel+ protocols in the case where $k = 16$, $k_1 = k_2 = 4$, $L = 6$, $d = |L|_4 = 2$. In this case, $\nu_1 = \nu_2 = \frac{4}{2}$, and hence $\tau_1 = 2 * 6 + 4$, $\tau_2 = 2 * 6 + 4$, and $\tau = 16 + 4 * 16 = 80$. Note that the actor performs $\frac{\tau_1}{k_1} = 4$ actor transmission cycles of length k_1 to train the sensors on their macrocorona, and additional $\frac{\tau_2}{k_2} = 4$ actor transmission cycles of length k_2 per macrocorona to train them on their microcorona.

In general, with respect to the performance of the two-level protocols, one has:

THEOREM 3. *Fixed L, d, k, k_1, and k_2, with $k = k_1 k_2$ and $L \geq d \geq max\{(L, k_1), (L, k_2)\}$, letting ν_1 and ν_2 be, respectively, the numbers of sensor sleep/wake transitions required to train a sensor on k_1 macrocoronas and k_2 microcoronas, the two-level protocols require $\nu = \nu_1 + \nu_2$ sleep/wake transitions and $\omega = (\nu_1 + \nu_2)d$ overall sensor awake time. Moreover, the total time for training is $\tau = \tau_1 + \tau_2 k_1$, where τ_1 and τ_2 must be the upper bounds on the total time required by the flat protocol adopted on each level.*

Note that, by Theorems 1 and 2, tight bounds on the values of τ_1 and τ_2 can be derived only when $d = (L, k)$ and $d = |L|_k$. In all other cases, the total time of each level is derived from $\tau = \nu L + k$, setting ν equal to the upper bound given in Theorems 1 and 2. For example, consider the TwoLevel– protocol and assume $d = |L|_{k_1}$, and $(L, k_2) \leq d < |L|_{k_2}$. Then, $\tau_1 = \left(\left\lfloor \frac{k_1}{|L|_{k_1}} \right\rfloor + 1\right) L + k_1$ and $\tau_2 = \left(\frac{k_2}{(L,k_2)} + \left|\frac{1}{L'}\right|_{k'}\right) L + k_2$, where $k' = \frac{k_2}{(L,k_2)}$ and $L' = \frac{L}{(L,k_2)}$.

Next, the worst case performance of the Flat protocol is compared with that of the corresponding TwoLevel protocol when the same value of L and d are used, $\frac{k_1}{(L,k_1)} \neq 1$, $\frac{k_2}{(L,k_2)} \neq 1$, and $(L, k_2) \neq 1$. Note that to satisfy the constraints of both the Flat and TwoLevel protocols, d must vary between $(L, k) = (L, k_1)(L, k_2)$ and $\min\{k_1, k_2\}$, and L must be greater than k. When $d = (L, k)$, by Theorem 2, the number of transitions is at most $\frac{k_1}{(L,k_1)} + \frac{k_2}{(L,k_2)}$ for TwoLevel and is at least $\frac{k}{(L,k)}$ for Flat. Since $(L, k) = (L, k_1)(L, k_2)$, one has $\frac{k_1}{(L,k_1)} + \frac{k_2}{(L,k_2)} < \frac{k_1}{(L,k_1)} \frac{k_2}{(L,k_2)} = \frac{k}{(L,k_1)(L,k_2)} = \frac{k}{d}$. Similarly, TwoLevel beats Flat when $d = \min\{k_1, k_2\}$. Indeed letting $d = k_1 = \min\{k_1, k_2\}$, TwoLevel requires at most $1 + \frac{k_2}{(L,k_2)} < k_2$ sleep/wake transitions, while Flat needs at least $\frac{k}{d} = k_2$ transitions. Since in both protocols the number of transitions decreases when d increases, TwoLevel beats Flat when $(L, k) \leq d \leq \min\{k_1, k_2\}$. Finally it is easy to see that both the overall sensor awake time and the total time for training of Flat are larger than those of TwoLevel.

5. EXPERIMENTAL TESTS

In this section, the worst and average performance of the two-level corona training protocols are experimentally tested. The algorithms were written in C++ and the experiments

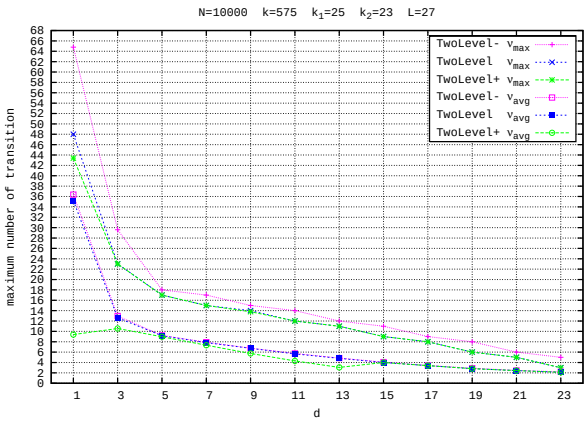

Figure 10: *Number of transitions when* $k = 575$, $k_1 = 25$, $k_2 = 23$, $L = 27$, *and* $1 \leq d \leq 23$.

were run on an AMD Athlon X2 4800+ with 2 GB RAM. In the simulation, each corona has a unit width. There are $N = 10000$ sensors uniformly distributed within a circle, centered at the actor, having radius $\rho = k$. Precisely, the polar coordinates of each sensor are generated choosing at random two real numbers. The first one, uniformly distributed between 0 and k, represents the radial coordinate of the sensor, that is, its distance from the actor. The second number, uniformly distributed between 0 and 2π, represents the angular coordinate of the sensor, that is, the positive angle required to reach the sensor from the polar axis. In the experiments, both the worst and average number of transitions, denoted by $\nu_{\max}$ and ν_{avg}, as well as both the worst and average overall sensor awake time, $\omega_{\max}$ and ω_{avg}, are evaluated. Such average values are obtained by summing up the values for each single sensor and then dividing by the number of sensors. Moreover, the total time τ, which measures the time required to terminate the whole training process, is evaluated.

Recall that TwoLevel–, TwoLevel, and TwoLevel+ denote, respectively, the protocol when Flat–, Flat, and Flat+ are employed on each single level. In the simulations, the number k of coronas is fixed to 575 while k_1 and k_2 are fixed to 25 and 23, respectively. The length L of the sensor sleep-awake cycle is fixed to 27 and the sensor awake period d varies, with a step of 2, between $\max\{(L, k_1), (L, k_2)\} = 1$ and $\min\{k_1, k_2\} = 23$. The results are averaged over 3 independent experiments.

Figures 10, 11, and 12 plot both the average and worst case performance of ν, ω, and τ. As explained in the previous section, one can easily derive the worst case performance of the two-level protocols in Figure 10 from the worst case performance of the one-level protocols. For example, when $d = (L, k_1) = (L, k_2) = 1$, TwoLevel– requires $\nu = 65$ sleep/wake transitions because the Flat– protocol requires $\nu_1 = \frac{k_1}{(L,k_1)} + \left|\frac{1}{L}\right|_{k_1} = 25 + 15 = 38$ transitions when $k_1 = 25$ and $\nu_2 = \frac{k_2}{(L,k)} + \left|\frac{1}{L}\right|_{k_2} = 23 + 6 = 19$ transitions when $k_2 = 23$.

Figure 11 shows the awake times $\omega_{\max} = \nu_{\max} d$ and $\omega_{\text{avg}} =$

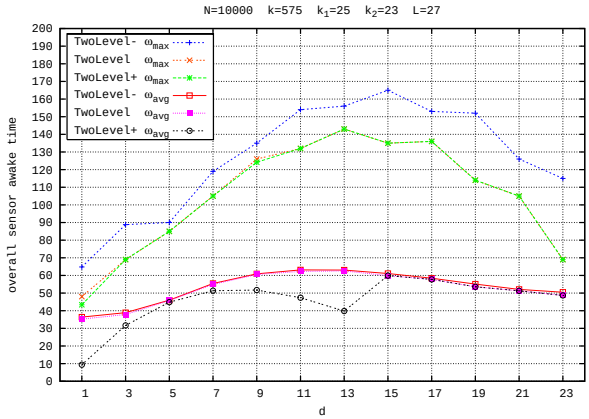

Figure 11: ***Overall sensor awake time when*** $k = 575$, $k_1 = 25$, $k_2 = 23$, $L = 27$, ***and*** $1 \leq d \leq 23$.

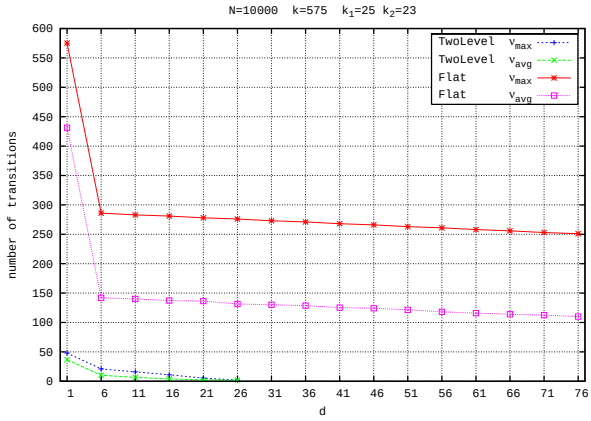

Figure 13: ***Comparing the number of transitions between Flat and TwoLevel.***

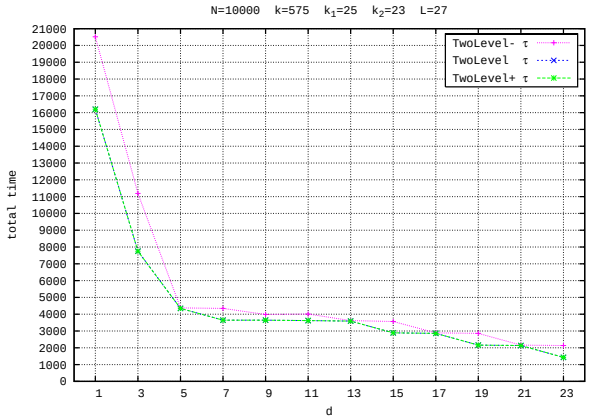

Figure 12: ***Total time for training when*** $k = 575$, $k_1 = 25$, $k_2 = 23$, $L = 27$, ***and*** $1 \leq d \leq 23$.

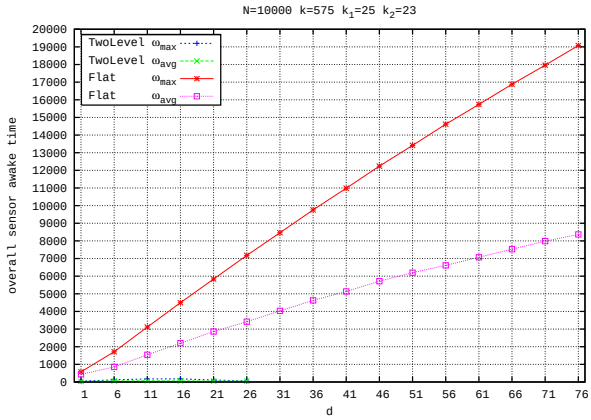

Figure 14: ***Comparing the overall sensor awake time between Flat and TwoLevel.***

$\nu_{\text{avg}} d$. Note that when $d = (L, k) = 1$ the overall awake time is minimum, although the number of transitions is maximum, and TwoLevel+ reaches the maximum gain with respect to the other algorithms, in both the worst and average cases. For $d = 13$, the TwoLevel+ protocol registers on both ν ad ω a local minimum.

Figures 13, 14, and 15 are devoted to compare the behaviour of the Flat and TwoLevel protocols. As before, $k = 575$, $k_1 = 25$, and $k_2 = 23$. The length L of the sensor sleep-awake cycle is fixed to 27 and 577 for TwoLevel and Flat, respectively. Note that both such values of L are the smallest possible choices for the two protocols, because Flat requires L larger than k and TwoLevel needs L larger than $\max\{k_1, k_2\}$. The sensor awake period d varies, with a step of 5, between $(L, k) = 1$ and 76. The results are averaged over 3 independent experiments.

As expected for $1 = (L, k) \leq d \leq 23 = \min\{k_1, k_2\} = 23$, the TwoLevel protocol always significantly beats the Flat protocol. Note that, in contrast to Flat, TwoLevel cannot be employed when $d \geq 23$. Observe that $\nu = O(1)$ can be achieved by both the Flat and TwoLevel protocols in correspondence of $d = \Theta(k)$ and $d = \Theta(\sqrt{k})$, respectively, leading therefore to a big difference in the values of ω and τ.

Although two different values of L are used in the experiments, $L = 577$ could be also used for the TwoLevel protocol. In such a case, since $|577|_{25} = |27|_{25}$, the number of transitions ν_1 on the macrocoronas would remain the same, whereas the number of transitions ν_2 on the microcoronas would become slightly larger because $|577|_{23} = 2 < |27|_{23} = 4$. Therefore, $\nu = \nu_1 + \nu_2$ would increase at most by $\lceil \frac{23}{2} \rceil - \lceil \frac{23}{4} \rceil = 6$ with respect to that shown in Figure 13. Obviously the total time τ for TwoLevel with $L = 577$ would dramatically increase with respect to that illustrated in Figure 15, but it would still remain below that of Flat.

6. CONCLUDING REMARKS

In this work new corona training protocols have been proposed which employ the asynchronous model originally presented in [3, 16] and are lightweight in terms of the number of sleep/wake transitions and overall sensor awake time for

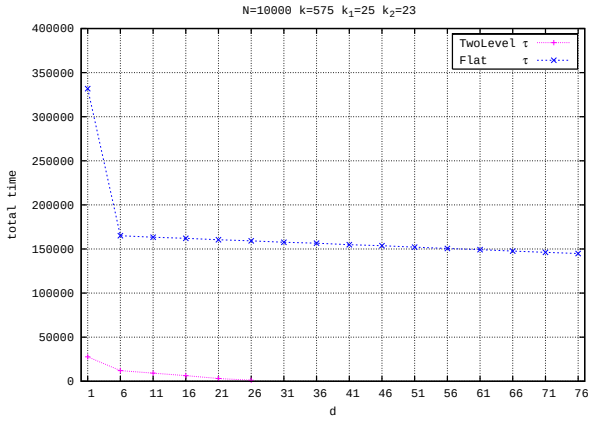

Figure 15: ***Comparing the total time for training between Flat and TwoLevel.***

training. Experimental tests showed that, among the various protocol variants, TwoLevel+ has the best performance.

The results presented in this paper show that the protocols are flexible, in the sense that their parameters can be properly tuned. For instance, fixed the number $k = k_1 k_2$ of coronas, one can decide the optimal values of d and L so as to minimize the number of sleep/wake transitions and/or the overall awake time per sensor. Conversely, one can fix the desired number of sleep/wake transitions, and then select suitable values of d and L.

However, several questions still remain open. In the present paper, a single fixed actor was assumed for training the sensors. Can several fixed actors, knowing about each other, collaboratively train sensors better? Can these protocols be adapted to the case of one or more mobile actors? Perhaps coronas and wedges can be defined differently so that static sensors reuse previous knowledge combined with new signals possibly containing location of actor?

7. REFERENCES

[1] I.F. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad Hoc Networks*, 2, 351–367, 2004.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramanian, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4), 393–422, 2002.

[3] F. Barsi, A.A. Bertossi, F. Betti Sorbelli, R. Ciotti, S. Olariu, and M.C. Pinotti. Asynchronous training in wireless sensor networks. *Proc. 3rd AlgoSensors*, Wroclaw, Poland, July 2007.

[4] A.A. Bertossi, S. Olariu, and M.C. Pinotti. Efficient training of sensor networks. *Proc. 2nd AlgoSensors*, Venice, Italy, July 2006.

[5] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh. Optimal energy-aware clustering in sensor networks. *Sensors*, 2, 258–269, 2002.

[6] H. Griffin, *Elementary Theory of Numbers*, McGraw Hill, New York, 1954.

[7] K. Langendoen and N. Reijers. Distributed localization algorithm. In *Embedded Systems Handbook*, R. Zurawski (Editor), CRC Press, Boca Raton, FL, 2004.

[8] J.J. Lee, B. Krishnamachari, and C.C. Jay Kuo. Impact of heterogeneous deployment on lifetime sensing coverage in sensor networks. *Proc. IEEE SECON*, 2004.

[9] V. Mhatre and C. Rosenberg. Design guidelines for wireless sensor networks: communication, clustering and aggregation. *Ad Hoc Network*, 2, 45–63, 2004.

[10] S. Olariu, M. Eltoweissy, and M. Younis. ANSWER: autonomous networks sensor systems. *Journal of Parallel and Distributed Computing*, 67, 114–126, 2007.

[11] S. Olariu, A. Waada, L. Wilson, and M. Eltoweissy. Wireless sensor networks leveraging the virtual infrastructure. *IEEE Network*, 18(4), 51–56, 2004.

[12] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Comm.*, 7(5), 16–27, 2000.

[13] K. Sohrabi et al. Methods for scalable self-assembly of ad hoc wireless sensor networks. *IEEE Transactions on Mobile Computing*, 3(4), 317–331, 2004.

[14] A. Waada, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones. Training a wireless sensor network. *Mobile Networks and Applications*, 10(1), 151–168, 2005.

[15] B. Warneke, M. Last, B. Leibowitz, and K. Pister. SmartDust: communicating with a cubic-millimeter computer. *IEEE Computer*, 34(1), 44–51, 2001.

[16] Q. Xu, R. Ishak, S. Olariu, and S. Salleh. On asynchronous training in sensor networks. *Proc. 3rd Intl. Conf. on Advances in Mobile Multimedia*, K. Lumpur, September 2005.