Mobile Information Systems 3 (2007) 233–249
IOS Press

Hybrid training with binary search protocol for wireless sensor networks

Ruzana Ishak[a,*], Qingwen Xu[b], Stephan Olariu[b] and Shaharuddin Salleh[c]
[a]*Department of Science* (*Mathematics*), *UTM CityCampus, 54100 Kuala Lumpur, Malaysia*
[b]*Department of Computer Science, Old Dominion University, Norfolk, Virginia 23529, USA*
E-mail: {*xu_q,olariu*}*@cs.odu.edu*
[c]*Department of Mathematics, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia*
E-mail: ss@mel.fs.utm.my

Abstract. Locationing problem in Wireless Sensor Networks(WSNs) can be viewed as a general distributed sensor problem. It is with sensors that can discover other nodes or estimate ranges between nodes, that serve as position references. In this paper, we show that sensors acquire coarse-grain location awareness by the training protocol. The training protocol which hybrids the synchronization and training procedure. In this protocol, synchronization and training are combined into one scheme. The sink node sends two beacons in each slot instead of one. In the training, sensor searching for its location using a binary search scheme. Our simulation results shown less number of cycles needed for training.

Keywords: Wireless sensor networks, self-organization, dynamic coordinate system, hybrid-training protocols

1. Introduction

Determining the location of devices or objects is important in many applications. For outdoor environments, the most well-known positioning system is the Global Positioning System (GPS) [6,12], that uses 24 satellites set up by the US. Department of Defense to enable global three-dimensional positioning. GPS has two levels of accuracy: stand positioning service (SPS) and precise positioning service (PPS). The accuracy provided by GPS is around 20 to 50 m. GPS is not suitable for wireless sensor networks for several reasons. First, it is not available in an indoor environment because satellite signals cannot penetrate buildings. Second, sensor networks have stringent energy constraints, which require special design.

Much work has been devoted recently to positioning and location tracking in the area of wireless ad hoc and sensor networks. Location information can be used to improve the performance of wireless networks and to provide new types of services. For example, it can facilitate routing in a wireless ad hoc network to reduce routing overhead. This is known as geographic routing [9,12]. Through location-aware network protocols, the number of control packets can be reduced. Service providers can also use location information to provide some novel location-aware or follow-me services. For example, the navigation system based on GPS. A user can tell the system his destination and the system will guide him there. Phone systems in an enterprise can exploit locations of people to provide *follow-me* services.

*Corresponding author. Tel.: +601 2298 2650; Fax: +603 2693 4844; E-mail: ruzanaishak@yahoo.com.

Other types of location-based services include *geocast* [8,10,12], by which a user can request to send a message to a specific area, and *temporal geocast*, by which a user can request to send a message to a specific area at specific time. In contrast to traditional multicast, such messages are not targeted at a fixed group of members, but rather at members located in a specific physical area.

However, the random deployment of sensors in wireless sensor networks implies that the sensors are initially unaware of their exact 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. It follows that the sensors have to learn either their exact geographic location or else a coarse-grain approximation of their location. The former task is referred to as *localization*. Wadaa et al. [13,14] and Xu et al. [11] proposed to refer to the task of endowing individual sensors with coarse-grain location awareness as *training*.

In this paper, we show that sensors acquire coarse-grain location awareness by the training protocol that imposes a coordinate system onto the network. We propose a simple lightweight and self-organization training algorithm to train the sensors learn their coarse-grain location. The training protocol combined the synchronization and training procedure into one scheme. We provide in Section 2 on how the dynamic system is proposed and in Section 3 the background of our training protocol in which the sensors are in sleep-awake cycle before the training started. Section 4 discusses the details of the algorithm of hybrid training protocol. In Section 4.2 we invite the reader to focus on the examples of our training protocol. We proof our training protocol by the simulation results in Section 5. Concluding remarks in Section 6 complete this paper.

2. Dynamic coordinate system

The process of endowing individual sensor nodes with coarse-grain location awareness leads this paper naturally to the concept of dynamic coordinate system. The dynamic system is a variant of the well known polar coordinate system adapted to our needs.

The dynamic coordinate system in polar forms is represented by the *coronas* for the range and the *wedges* for angles of coverage. This is illustrated by Fig. 1.

In Fig. 1, $\theta(x)$ is a polar angle corresponding to x and $\rho(x)$ is a polar distance corresponding to x. Localization means determining for all x in the plane the ordered pair $(\theta(x), \rho(x))$.

Discretize the space by considering k ranges for ρ, namely $r_1, r_2, \ldots, r_k$ such that $0 < r_1 < r_2 < \ldots < r_k = R$ and angular ranges $w_1, w_2, \ldots, w_m$ such that $0 < w_1 < w_2 < \ldots < w_m = 2\pi$. Thus for a given x, determine an approximation of its exact location in the form of a circular sector (see Fig. 2).

3. Synchronous vs. asynchronous training

Due to the extreme power limitation and lack of means for changing or recharging batteries, energy conversation is the most important design requirement for the WSN. It is concluded that the *idle listening* is the major source of power waste [2]. Leaving the radio transceiver on for long periods of time will be significantly reducing the longevity of the WSN. Hence, the sensor radios have to work on a low duty cycle, called a *sensor cycle*, in which the sensors will be awake only a short period of time (referred as *awake interval*) and sleep for the rest of the time.

Referring to Fig. 3, the sensor cycle is on length L (slots) [Note: slots are defined arbitrarily but a good practical value is 675 microsecond per slot]: the sensor is in sleep mode for $L - d$ slots and is awake for

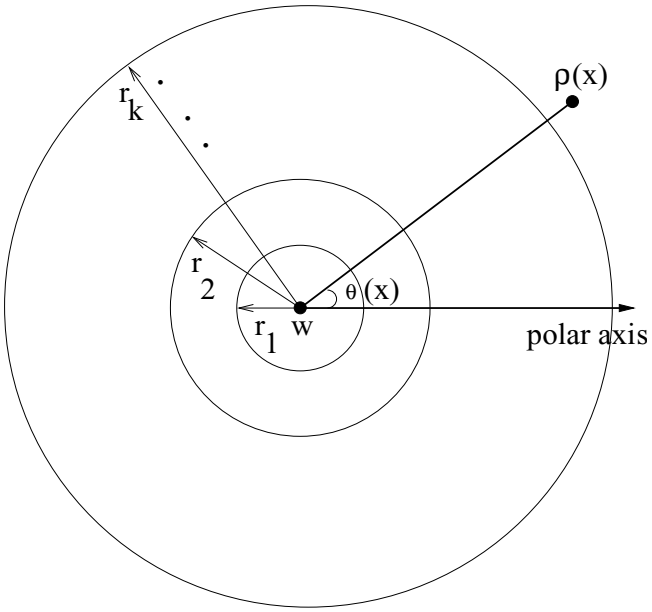

Fig. 1. The polar coordinate system.

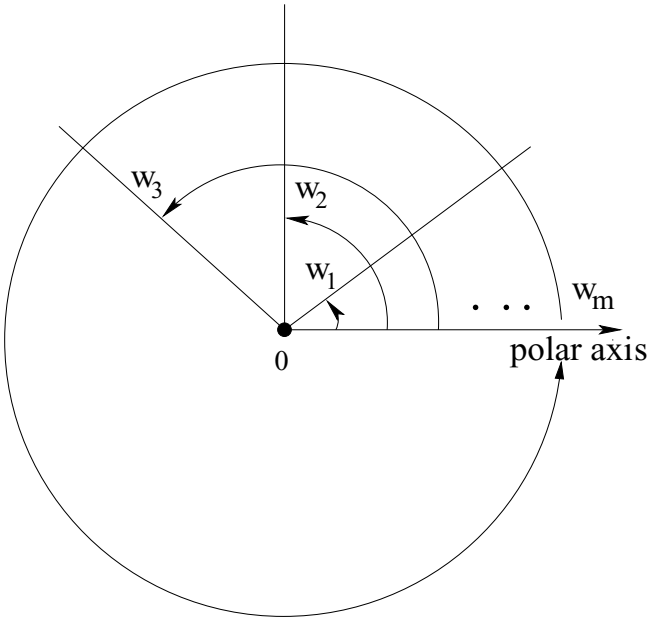

Fig. 2. Illustrating localization .

d slots. We assume the value of L is fixed during the network lifetime. However, the value of d and the beginning of the awake interval can be varied in different protocols.

In order to train the sensors in the network, the sink broadcasts training beacons. Assuming that k coronas have to be established, the sink *transmission cycle* involves k broadcasts in each k*-cycle*. Thus,

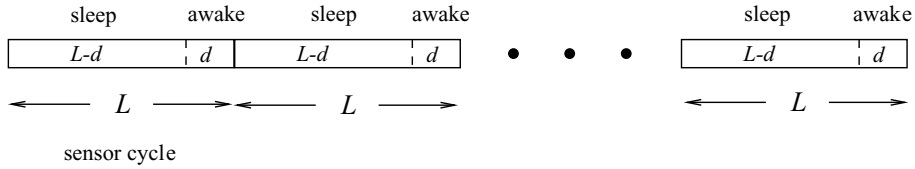

Fig. 3. Illustrating the sensor sleep-awake cycle.

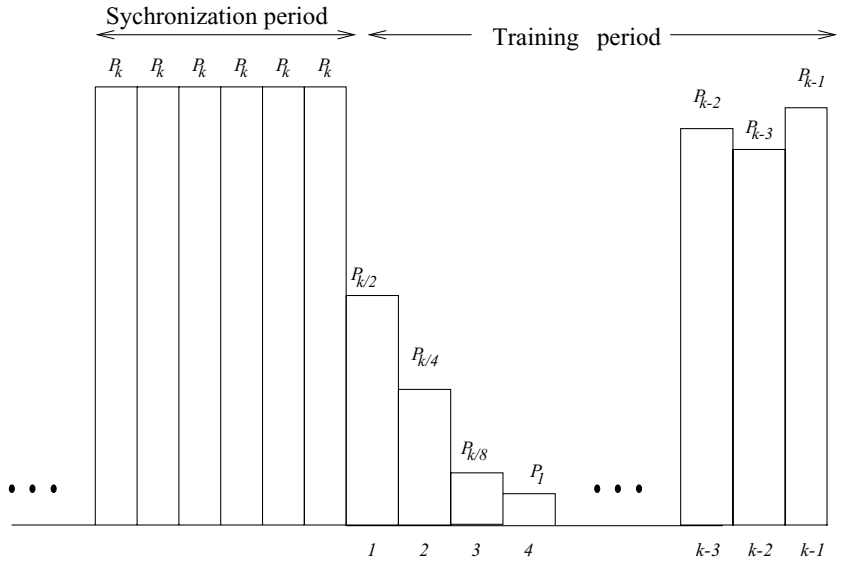

Fig. 4. Synchronization and training.

each $k - cycle$ consists of k slots encountered as $s_1, s_2, \ldots, s_k$ where slot s_j,$(1 \leqslant j \leqslant k)$, is intended to train the sensors in corona $k - j + 1$. Indeed, the power level that the sink uses in slot j distinguishes between sensors in slots $k + j - 1$ and $k - j$. We assume the sink can transmit beacons at different power level. At the highest power level (P_k), the beacon can be received by the sensors in the outmost corona s_k. At the lowest power level (P_1), the beacon can only be received by the sensors in the corona s_1. The sink repeats the transmission cycle (referred as the *sink cycle*) in case that some sensors miss the previous training beacons. To distinguish between sink cycles and sensor cycles, we shall refer to sink cycles as k*-cycles* and to the sensor cycles as s*-cycles*.

Since the sensors are not aware of the beginning time of the training, a training protocol relies on the sink to synchronize the sensors. Recall the sensors wake up at random according to their internal clock and alternate between sleep and awake in each s*-cycle*. The sink broadcasts a beacon at the highest power level for L slots to synchronize the sensors. We can limit each sensor be only awake one slot per s*-cycle* to save energy. It guarantees that every sensor in the range receives the beacon at least one time in the synchronization period. After the synchronization period, the sink starts training the sensors. Figure 4 illustrates. The sink cycle in the protocol proposed by Wadaa et al. in [13] is referred as the *synchronous training* protocol and requires that the sensors know the training start time before training.

The sensors can be trained without a synchronization period. An *asynchronous training* protocol was proposed in [11]. The sink cycle in the asynchronous training protocol is illustrated in Fig. 5. In this type of protocols, the values of k and L are critical for energy saving. The protocol requires that the

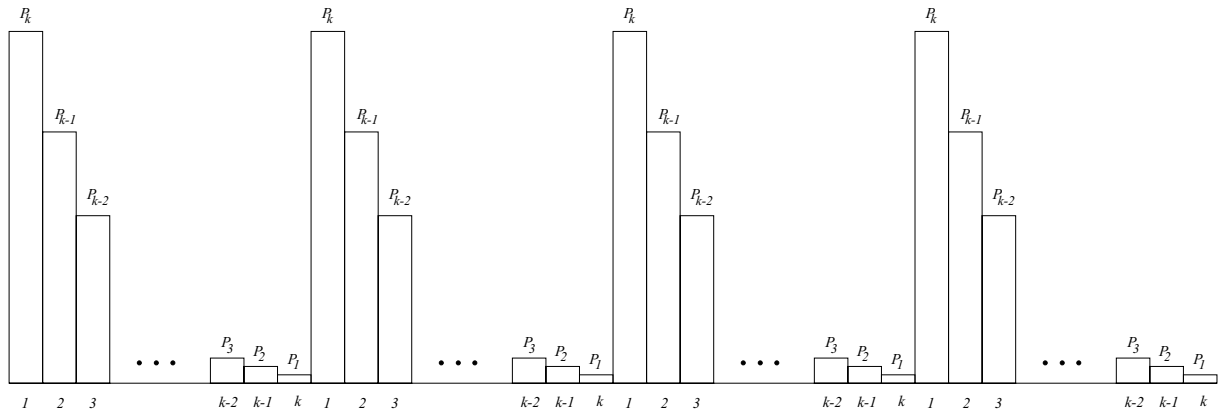

Fig. 5. Illustrating the sink transmission cycles.

sensors know the value of k in advance in order to minimize energy consumption.

The sensor determines that it belongs to corona s_1 by receiving a P_1 beacon. And the sensor determines that it belongs to corona s_i (where $i \neq 1$) by:

1. Receiving a P_{s_i} beacon
2. Failing to receive a $P_{s_{i-1}}$ beacon

The way the *k-cycle* is set up, in slot $i(1 \leqslant i \leqslant k)$ the sink transmitting at a power level P_{s_i}, where $s_i = k - i + 1$.

4. Hybrid training with binary search protocol

In this section, we propose a hybrid training with binary search protocol. In this protocol, synchronization and training are combined into one scheme. The sink node sends two beacons in each slot instead of one. The first beacon (referred as *synchronization beacon* or *SYNC* for short, is sent with full power for sensor synchronization. It contains the total corona number (k) and the current corona number (s). The second beacon (referred as *training beacon* or *TRAIN* for short, is sent at successively lower power levels for sensor training. The sink cycle is illustrated in Fig. 6.

We assume that a sensor is awake for one slot in every k-*cycle*. Before the training begins, sensors wake up at random until they receive a SYNC beacon. Thus, the probability of waking up in slot j is $\frac{1}{k}$. By receiving the beacon in a given slot $j, (1 \leqslant j \leqslant k)$, the sensor determines the following:

– Slot boundaries
– Synchronization to the k-*cycles*
– The value of k
– The beginning and end of the k-*cycle*

Consider a sensor in corona $i, (1 \leqslant i \leqslant k)$ that wakes up at random in slot j of some k-*cycle*, as showed in Fig. 7.

If the sensor receives the synchronization beacon but not the training beacon in slot j, the sensor also knows that its corona is in range of $[s, k]$. If the sensor receives both the synchronization beacon and the training beacon, then it knows its corona is in range of $[1, s-1]$. In order to determine its corona number, the sensor can employ a binary search scheme on its corona number range.

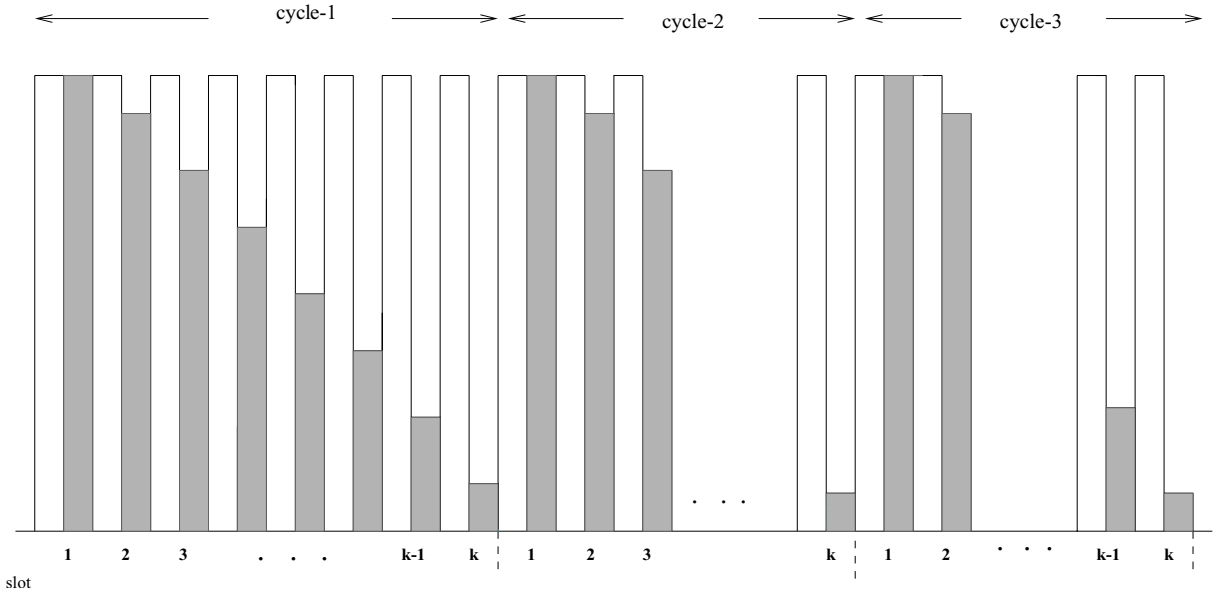

Fig. 6. Illustrating the hybrid training sink cycles.

Fig. 7. Illustrating the position of slot number and corona number.

In the binary search scheme, the sensor starts to listen in the middle of its range and computes its new range based on whether the training beacon is received. Assume its corona number range is *[low, high]*, the sensor starts to listen at slot $\lceil \frac{low+high}{2} \rceil$, if the training beacon is received, its new range is $[\lceil \frac{low+high}{2} \rceil, low]$. Otherwise its new range is $[high, \lceil \frac{low+high}{2} \rceil + 1]$. The procedure continues until its range boundaries are equal. The boundary gives the sensors corona number. The binary search scheme introduced here slightly differs from the traditional binary search algorithm. First, the scheme starts at an arbitrary position instead of the middle of the range. Second, the scheme ends when the searching range length reaches 1 instead of finding a particular value.

Note that in slot j the sink is transmitting to the intention of corona i, where $i = k + 1 - j$. As shown in Fig. 8, for case 1: if $i < k + 1 - j$ the sensor receives the TRAIN beacon. It then goes to sleep and to wake up when the corona number $\lceil \frac{(k+1-j)+1}{2} \rceil$ is trained in the next k-*cycle*.

Clearly, the slot being addressed in corona $\lceil \frac{(k+1-j)+1}{2} \rceil$ is:

$$k + 1 - \left\lfloor \frac{(k+1-j)+1}{2} \right\rfloor = k + 1 - \left\lfloor \frac{k-j}{2} + 1 \right\rfloor$$
$$= k - \left\lfloor \frac{(k-j)}{2} \right\rfloor$$

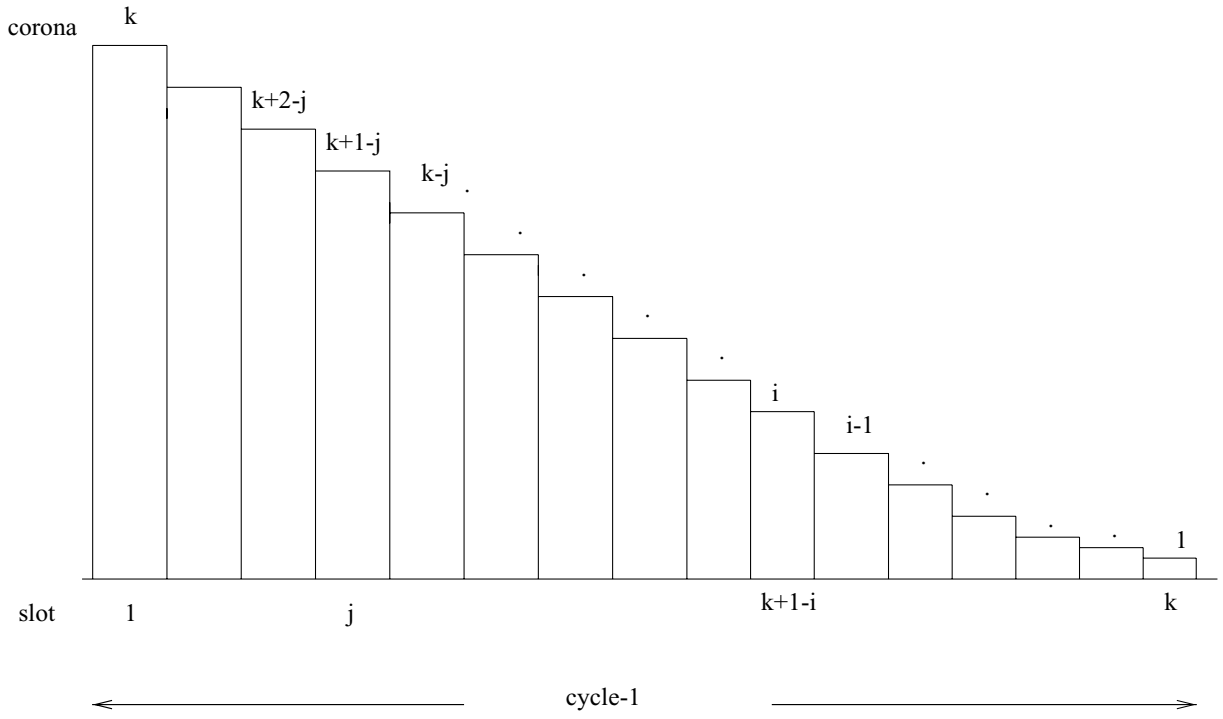

Fig. 8. Illustrating the sink transmission cycles.

$$= \left\lceil \frac{k+j}{2} \right\rceil. \tag{1}$$

This corresponds to the intuition idea that the sensor proceeds by binary search in the range $[j+1, k]$.

On the other hand, for case 2: if $i > k+1-j$, then the sensor does not receive the TRAIN beacon. The sensor goes to sleep and will wake up again when the corona number $\frac{k+(k+2-j)}{2}$ is trained in the next k-*cycle*. It is easy to confirm that the slot number assigned to the corona $\frac{k+(k+2-j)}{2}$ is:

$$\begin{aligned} k+1-\left\lfloor \frac{k+(k+2-j)}{2} \right\rfloor &= k+1-\left\lfloor \frac{2k+2-j}{2} \right\rfloor \\ &= k+1-\left\lfloor \frac{2k-j}{2}+1 \right\rfloor \\ &= k-\left\lfloor k-\frac{j}{2} \right\rfloor \\ &= -\left\lfloor -\frac{j}{2} \right\rfloor \\ &= \left\lceil \frac{j}{2} \right\rceil. \end{aligned} \tag{2}$$

4.1. Training with binary position search

Let $A(i)$ be a random variables that counts the number of k-*cycles* needed on the average to train a sensor in a fixed corona i. For case 1, let $A_1(i)$ be the expected number of k-cycles needed and it receives

the beacon in the first-cycle. Let P_j be the probability that the sensor wakes up in slot j, we can write:

$$\begin{aligned} A_1(i) &= \sum_{j=1}^{k+1-i} P_j\Big(1+\lceil \log_2 (k+1-j)\rceil\Big) \\ &= \sum_{j=1}^{k+1-i} \frac{1}{k}\Big(1+\lceil \log_2 (k+1-j)\rceil\Big) \\ &= \frac{k+1-i}{k} + \sum_{j=1}^{k+1-i} \frac{\lceil \log_2 (k+1-j)\rceil}{k} \\ &= \frac{k+1-i}{k} + \frac{1}{k}\sum_{t=1}^{k}\lceil \log_2 t\rceil - \frac{1}{k}\sum_{t=1}^{i-1}\lceil \log_2 t\rceil. \end{aligned} \tag{3}$$

Our evaluation of $A_1(i)$ relies on the following technical lemma.

Lemma 1. $\lfloor \log_2 (k-j)\rfloor + 1 = \lceil \log_2 (k-j+1)\rceil$

Proof: For $\forall n \in \mathbf{N}$ we have:

$$\lfloor \log_2 n\rfloor + 1 = \lceil \log_2 (n+1)\rceil$$

$\exists! K \in \mathbf{N}$ with $2^K \leqslant n < 2^{K+1}$ for some $K \in \mathbf{N}$. We have $K \leqslant \log_2 n < K+1$ and if $K = \lfloor \log_2 n\rfloor$ and $K \leqslant \log_2 n < \log_2 (n+1) \leqslant K+1$, consequently,

$$\lceil \log_2 (n+1)\rceil = K+1.$$

Thus,

$$\lfloor \log_2 n\rfloor + 1 = K+1 = \lceil \log_2 (n+1)\rceil.$$

For case 2, let $A_2(i)$ be the expected number of k-cycles needed and the sensor does not receive the beacon in the first-cycle. Let P_j be the probability that the sensor wakes up in slot j, we can write:

$$\begin{aligned} A_2(i) &= \sum_{j=k+2-i}^{k} P_j\Big(1+\lceil \log_2 (j-1)+1\rceil\Big) \\ &= \sum_{j=k+2-i}^{k} \frac{1}{k}\Big(1+\lceil \log_2 j\rceil\Big) \\ &= \frac{k-(k+2-i)+1}{k} + \sum_{j=k+2-i}^{k} \frac{\lceil \log_2 j\rceil}{k} \\ &= \frac{i-1}{k} + \sum_{t=1}^{k} \frac{\lceil \log_2 t\rceil}{k} - \sum_{t=1}^{k+1-i} \frac{\lceil \log_2 t\rceil}{k}. \end{aligned} \tag{4}$$

Thus, we can write $A(i)$ as:

$$A(i) = A_1(i) + A_2(i) \tag{5}$$

where

$$A_1(i) = \sum_{j=1}^{k+1-i} \Big(\frac{1}{k} + \lceil \log_2 (k+1-i) \rceil\Big)$$

and

$$A_2(i) = \sum_{j=k+2-i}^{k} \Big(\frac{1}{k} + \lceil \log_2 j \rceil\Big).$$

$$\begin{aligned}
A(i) &= \frac{k+1-i}{k} + \frac{1}{k}\sum_{t=1}^{k}\lceil \log_2 t\rceil - \frac{1}{k}\sum_{t=1}^{i-1}\lceil \log_2 t\rceil + \frac{i-1}{k} + \frac{1}{k}\sum_{t=1}^{k}\lceil \log_2 t\rceil - \frac{1}{k}\sum_{t=1}^{k+1-i}\lceil \log_2 t\rceil \\
&= \frac{k+1-i+(i-1)}{k} + \frac{1}{k}\Big(\sum_{t=1}^{k}\lceil \log_2 t\rceil - \sum_{t=1}^{i-1}\lceil \log_2 t\rceil + \sum_{t=1}^{k}\lceil \log_2 t\rceil - \sum_{t=1}^{k+1-i}\lceil \log_2 t\Big) \\
&= 1 + \frac{1}{k}\Big(\sum_{t=1}^{k}\lceil \log_2 t\rceil - \sum_{t=1}^{i-1}\lceil \log_2 t\rceil + \sum_{t=1}^{k}\lceil \log_2 t\rceil - \sum_{t=1}^{k+1-i}\lceil \log_2 t\rceil\Big) \\
&= 1 + \frac{2}{k}\sum_{t=1}^{k}\lceil \log_2 t\rceil - \frac{1}{k}\Big(\sum_{t=1}^{i-1}\lceil \log_2 t\rceil + \sum_{t=1}^{k+1-i}\lceil \log_2 t\rceil\Big).
\end{aligned} \tag{6}$$

Let $E[T]$ be the random variable describing the number of $k-cycles$ needed to train a sensor in the system, assuming that the total number of corona is k, the expected value $E[T]$ that we want to determine can be written as:

$$E[T] = \sum_{i=1}^{k} P(i)A(i). \tag{7}$$

$$\begin{aligned}
E[T] &= \sum_{i=1}^{k}\frac{1}{k} + \sum_{i=1}^{k}\sum_{t=1}^{k}\frac{2}{k^2}\lceil \log_2 t\rceil - \sum_{i=1}^{k}\sum_{t=1}^{i-1}\frac{1}{k^2}\lceil \log_2 t\rceil - \sum_{i=1}^{k}\sum_{t=1}^{k+1-i}\frac{1}{k^2}\lceil \log_2 t\rceil \\
&= 1 + \frac{2}{k}\sum_{t=1}^{k}\lceil \log_2 t\rceil - \frac{1}{k^2}\Big(\sum_{i=1}^{k}\sum_{t=1}^{i-1}\lceil \log_2 t\rceil - \sum_{i=1}^{k}\sum_{t=1}^{k+1-i}\lceil \log_2 t\rceil\Big) \\
&= 1 + \frac{2}{k}\sum_{t=1}^{k}\lceil \log_2 t\rceil - \frac{1}{k^2}\sum_{i=1}^{k}\sum_{t=1}^{i-1}\lceil \log_2 t\rceil - \frac{1}{k^2}\sum_{i=1}^{k}\sum_{t=1}^{k+1-i}\lceil \log_2 t\rceil.
\end{aligned} \tag{8}$$

Our evaluation of $E[T]$ relies on the following summations:

Claim 1. $\sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log_2 j\rceil = \sum_{i=1}^{k}\sum_{j=1}^{i}\lceil\log_2 j\rceil - \sum_{i=1}^{k}\lceil\log_2 i\rceil$.

Claim 2. $\sum_{i=1}^{k}\sum_{j=1}^{i}\lceil\log j\rceil = (k+1)\sum_{i=1}^{k}\lceil\log_2 i\rceil - \sum_{i=1}^{k} i\lceil\log_2 i\rceil$.

Proof:

$$\begin{aligned}\sum_{i=1}^{k}\sum_{j=1}^{i}\lceil\log_2 j\rceil &= \sum_{i=1}^{k}(k+1-i)\lceil\log i\rceil \\ &= \sum_{i=1}^{k}(k+1)\lceil\log_2 i\rceil - \sum_{i=1}^{k} i\lceil\log_2 i\rceil \\ &= (k+1)\sum_{i=1}^{k}\lceil\log_2 i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil. \end{aligned} \tag{9}$$

Claim 3. $\sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log_2 j\rceil = k\sum_{i=1}^{k}\lceil\log_2 i\rceil - \sum_{i=1}^{k} i\lceil\log_2 i\rceil$.

Proof:

$$\begin{aligned}\sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log_2 j\rceil &= \sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log_2 j\rceil - \sum_{i=1}^{k}\lceil\log_2 i\rceil \\ &= (k+1)\sum_{i=1}^{k}\lceil\log_2 i\rceil - \sum_{i=1}^{k} i\lceil\log_2 i\rceil - \sum_{i=1}^{k}\lceil\log i\rceil \\ &= k\sum_{i=1}^{k}\lceil\log_2 i\rceil - \sum_{i=1}^{k} i\lceil\log_2 i\rceil. \end{aligned} \tag{10}$$

Claim 4. $\sum_{i=1}^{k}\sum_{j=1}^{k+1-i}\lceil\log_2 j\rceil = \sum_{i=1}^{k}\sum_{j=1}^{i}\lceil j\rceil$.

Using the summation results, we have:

$$\begin{aligned}E[T] = 1 + \frac{2}{k}\sum_{t=1}^{k}\lceil t - \frac{1}{k^2}\rceil\Big(\sum_{i=1}^{k} k\lceil\log_2 i - \sum_{i=1}^{k} i\lceil\log_2 i\rceil + (k+1) \\ \Big(\sum_{i=1}^{k}\lceil\log_2 i\rceil - \sum_{i=1}^{k} i\lceil\log_2 i\rceil\Big)\Big) \\ = 1 + \frac{2}{k^2}\sum_{i=1}^{k} i\lceil\log_2 i\rceil - \frac{1}{k^2}\sum_{i=1}^{k}\lceil\log_2 i\rceil. \end{aligned} \tag{11}$$

Our final evaluation for $E[T]$ relies on the following theorems:

Theorem 1. Let (a_n) be an arbitrary sequence of variables. For $\forall n \in \mathbf{N}$, we have:

$$\sum_{i=1}^{n}(a_i) = na_n - \sum_{i=i}^{k-1} i(a_{i+1} - a_i).$$

Proof: This theorem is useful whenever the difference $a_{i+1} - a_i$ $(1 \leqslant i \leqslant n-1)$ is easy to evaluate.

$$\begin{aligned}\sum_{i=1}^{n}(a_i) &= na_n - ([a_2 + 2.a_3 + \ldots + (n-2).a_{n-1} + (n-1).a_n] - [a_1 + a_2 + \ldots \\ &\quad +(n-1).a_{n-1} + n.a_n] \\ &= na_n - (-a_1 - a_2 - \ldots - a_{n-1} + (n-1).a_n) \\ &= a_1 + a_2 + \ldots + a_n.\end{aligned}$$

Theorem 2. $\sum_{\alpha=1}^{k}(\lceil \log_2 \alpha \rceil) = k\lceil \log_2 k \rceil - 2^{\lceil \log_2 k \rceil} + 1.$

Proof: Replacing in Theorem 1 (a_n) by $\lceil \log_2 \alpha \rceil$ we have:

$$\sum_{\alpha=1}^{k}\lceil \log_2 \alpha \rceil = k\lceil \log_2 k \rceil - \sum_{\alpha=1}^{k-1} \alpha(\lceil \log_2 (\alpha+1) \rceil - \lceil \log_2 \alpha \rceil).$$

With $2^P \leqslant n < 2^{P+1}$ for some $P \in \mathbf{N}$. We have $P \leqslant \log_2 n < P+1$ and consequently,

$$\lceil \log_2 n \rceil = \begin{cases} P & n = 2^P \\ P+1 & 2^P < n < 2^{P+1} \end{cases}$$

The value of $\lceil \log_2 (\alpha+1) \rceil - \lceil \log_2 \alpha \rceil$ is either 0 or 1. The value is 1 whenever $i = 2^q$ for some $1 \leqslant q < \lceil \log_2 n \rceil$, and 0 everywhere else.

Thus,

$$\sum_{\alpha=1}^{k-1} \alpha(\lceil \log_2 (\alpha+1) \rceil - \lceil \log_2 \alpha \rceil) = \sum_{q=0}^{\lceil \log_2 k \rceil - 1} 2^q = 2^{\lceil \log_2 k \rceil} - 1.$$

The conclusion follows.

Theorem 3. $\sum_{i=1}^{k-1} i^2(\lceil \log_2 (i+1) \rceil - \lceil \log_2 i \rceil) = \frac{4^{\lceil \log_2 k \rceil - 1}}{3}$ With $2^\alpha \leqslant k < 2^{\alpha+1}$ for some natural number α.

Case 1: $k = 2^\alpha$

Notice that the difference $\lceil \log_2 (i+1) \rceil - \lceil \log_2 i \rceil$ Is non-zero only if $i = 2^\beta$ for some $\beta \leqslant \alpha - 1$. The value of the corresponding item is $2^{2\beta} = 4^\beta$. Thus, the value of the sum is:

$$\sum_{\beta=0}^{\alpha-1} 4^\beta = \frac{4^{\beta+1} - 1}{3} = \frac{4^\alpha - 1}{3} = \frac{4^{\lceil \log_2 k \rceil} - 1}{3}.$$

Case 2: $2^{\alpha} < k < 2^{\alpha+1}$ (we have $\lceil \log_2 k \rceil = \alpha + 1$)
As before, the term is 4^{β}. Thus the sum becomes:

$$\sum_{\beta=0}^{\alpha} 4^{\beta} = \frac{4^{\beta+1}-1}{3} = \frac{4^{\alpha+1}-1}{3} = \frac{4^{\lceil \log_2 k \rceil}-1}{3}.$$

As claimed.

Theorem 4. $\sum_{i=1}^{k} i\lceil \log_2 i \rceil = \frac{1}{2}\left[k^2\lceil \log_2 k \rceil + k\lceil \log_2 k \rceil - 2^{\lceil \log_2 k \rceil} + 1 - \frac{4^{\lceil \log_2 k \rceil}-1}{3}\right]$.

Proof: Using Theorem 1, we have:

$$\begin{aligned}\sum_{i=1}^{k} i\lceil \log_2 i \rceil &= k^2\lceil \log_2 k \rceil - \sum_{i=1}^{k-1} i((i+1)\lceil \log_2 (i+1) \rceil - i\lceil \log_2 i \rceil) \\ &= k^2\lceil \log_2 k \rceil - \sum_{i=1}^{k-1} i^2(\lceil \log_2 (i+1) \rceil - \lceil \log_2 i \rceil) - \sum_{i=1}^{k-1} i\lceil \log_2 (i+1) \rceil. \end{aligned} \tag{12}$$

Consequently

$$\sum_{i=1}^{k} i\lceil \log_2 i \rceil + \sum_{i=1}^{k-1} i\lceil \log_2 (i+1) \rceil = k^2\lceil \log_2 k \rceil - \sum_{i=1}^{k-1} i^2(\lceil \log_2 (i+1) \rceil - \lceil \log_2 i \rceil).$$

Eventually

$$\begin{aligned}&\sum_{i=1}^{k} i\lceil \log_2 i \rceil + \sum_{i=1}^{k-1} (i+1)\lceil \log_2 (i+1) \rceil - \sum_{i=1}^{k-1} \lceil \log_2 (i+1) \rceil \\ &\quad = k^2\lceil \log_2 k \rceil - \sum_{i=1}^{k-1} i^2(\lceil \log_2 (i+1) \rceil - \lceil \log_2 i \rceil).\end{aligned}$$

Note that:

$$\sum_{i=1}^{k-1} (i+1)\lceil \log_2 (i+1) \rceil = \sum_{i=2}^{k} i\lceil \log_2 i \rceil = \sum_{i=1}^{k} i\lceil \log_2 i \rceil$$

and

$$\sum_{i=1}^{k-1} \lceil \log_2 (i+1) \rceil = \sum_{i=2}^{k} \lceil \log_2 i \rceil = \sum_{i=1}^{k} \lceil \log_2 i \rceil.$$

We have

$$2\sum_{i=1}^{k} i\lceil \log_2 i \rceil = \sum_{i=1}^{k} \lceil \log_2 i \rceil + k^2\lceil \log_2 k \rceil - \sum_{i=1}^{k-1} i^2(\lceil \log_2 (i+1) \rceil - \lceil \log_2 i \rceil).$$

Applying Theorem 3 we have

$$\sum_{i=1}^{k} i\lceil \log_2 i\rceil = \frac{1}{2}\left[k^2\lceil \log_2 k\rceil + \sum_{i=1}^{k}\lceil log_2 i\rceil - \frac{4^{\lceil log_2 k\rceil}-1}{3}\right]$$

$$= \frac{1}{2}\left[k^2\lceil \log_2 k\rceil + k\lceil \log_2 k\rceil - 2^{\lceil \log_2 k\rceil}\rceil + 1 - \frac{4^{\lceil \log_2 k\rceil}-1}{3}\right]. \tag{13}$$

Now, we can evaluate the value of $E[T]$:

$$E[T] = 1 + \frac{2}{k^2}\sum_{i=1}^{k} i\lceil \log_2 i\rceil - \frac{1}{k^2}\sum_{i=1}^{k}\lceil \log_2 i\rceil$$

$$= 1 - \frac{1}{k^2}\Big(k\lceil \log_2 k\rceil - 2^{\lceil \log_2 k\rceil} + 1\Big) + \frac{2}{k^2}\Big(\frac{1}{2}\Big(k^2\lceil \log_2 k\rceil + \sum_{i=1}^{k}\lceil \log_2 i\rceil - \frac{4^{\lceil \log_2 k\rceil}-1}{3}\Big)\Big)$$

$$= 1 + \lceil \log_2 k\rceil - \frac{4^{\lceil \log_2 k\rceil}-1}{3k^2}. \tag{14}$$

4.2. Summary

We summaries the protocol performance with example as the following.

For $k = 8$, we assume that the sensor is awake in the first cycle when the sink is trained for the corona number 6, which is $k + 1 - j = 6$, then $j = 3$, and the actual corona number, $s = 5$. In this case, the sensor will not receives the beacon but it will learn to awake in the next *k-cycle* when the sink is training the corona number, $\lfloor \frac{k-j+2}{2}\rfloor = 3$.

The sensor also knows that its correct corona range is $[k + 1 - j, 1] = [6, 1]$. In cycle 2 as shown in Fig. 9, the sensor will awake at slot $k - \frac{k-j+2}{2} + 1 = 6$. At this slot, the sensor does not receive the beacon because it needs more power compare to the sensors in corona number 3. The searching range for the sensor is changed to [6,3]. The sensor continues learning to awake in cycle-3 when the sink is trained for the corona $\frac{6+3}{2} = 4$, which is in slot $k - 4 + 1 = 5$. The sensor does not receive the beacon. The searching range for the sensor is changed to [6,4]. In cycle-4, the sensor will awake when the corona $\frac{6+4}{2} = 5$ is trained, which is in slot $k - 5 + 1 = 4$. In this slot, the sensor receives the beacon and the searching range is [5,4]. Now, the boundaries are different by 1. The actual corona number will be the last slot when the sensor awakes and receives the beacon. It is in corona number 5.

In Example 2 as shown in Fig. 10 will do the same method of training for $k = 8$, $j = 7$ and $s = 6$. The expected of k-cycles needed as shown in the figure is 3.

5. Simulation

In our simulation, sensors are deployed uniformly at random in a field of size 700 m $\times$ 700 m. A sink node is placed in the middle of the field at (360, 360). The number k of coronas is 64 and each corona has a width of 10 meters. Hence, the length of a k-cycle is 64 time slots (a slot is 10 milliseconds). The sensors wake up at random to simulate the asynchronous effect.

For each sensor, the first wake-up time is generated uniformly at random in the interval $(0, L)$, where $L = 100$. Hence, each sensor will wake up at a random time between that first and the 100-nd time slot.

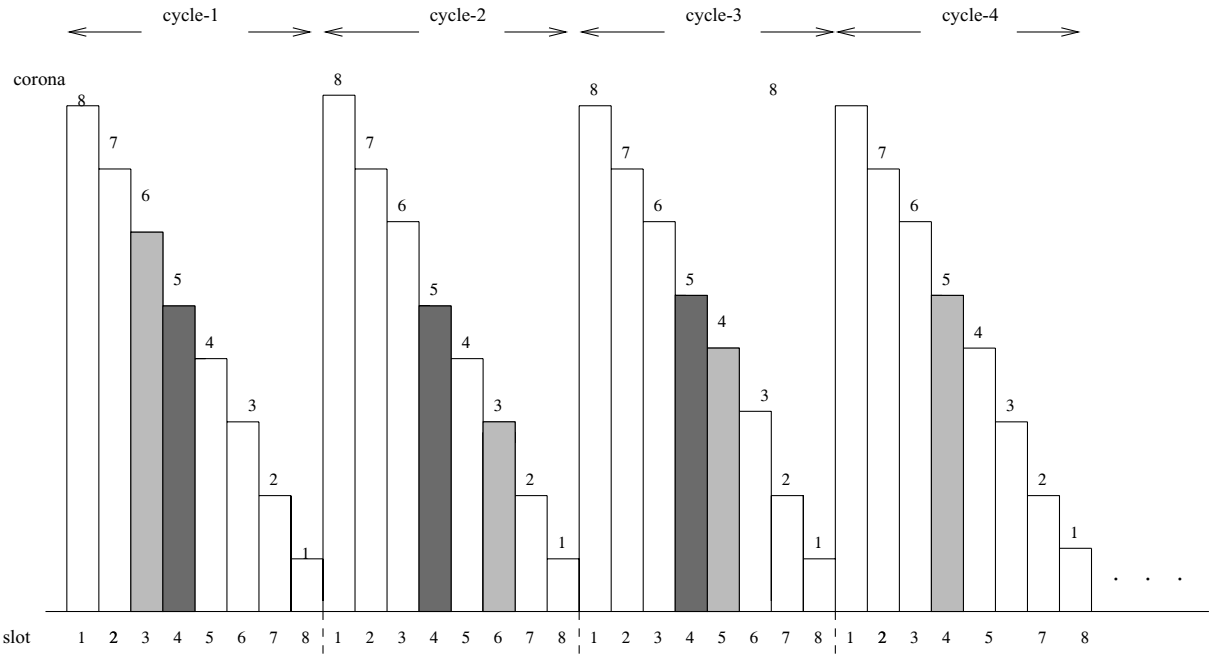

Fig. 9. Illustrating the example for hybrid training cycles.

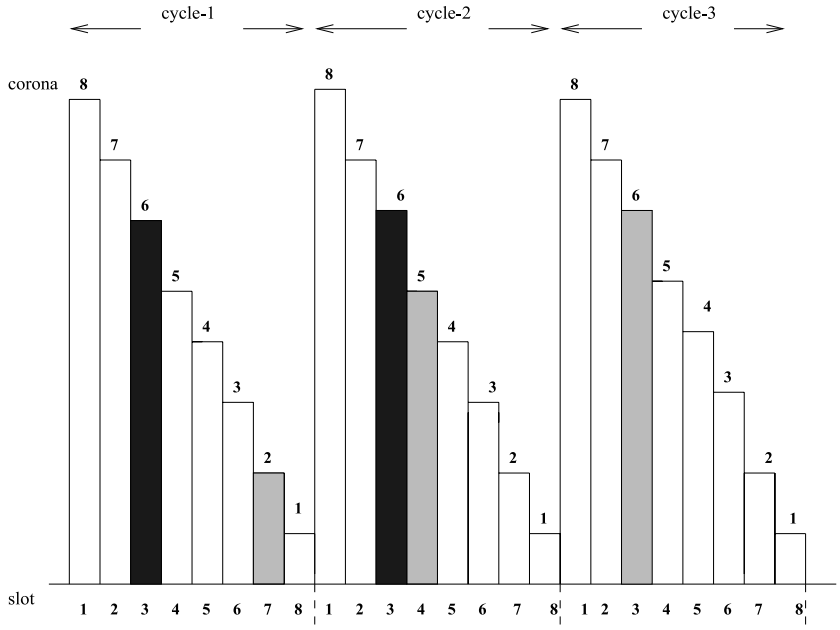

Fig. 10. An example for $k = 8, s = 6, j = 7$ in hybrid training cycles.

Figures 11 and 12 show the performance of the binary search scheme. We plotted the expected number of k-cycles needed to train the sensor node, E[T], for different k as shown in Fig. 11. As anticipated, the total time slot number needed for training is $T = L + k(\log_2 k - 1) + 1 = 149$, and our simulation

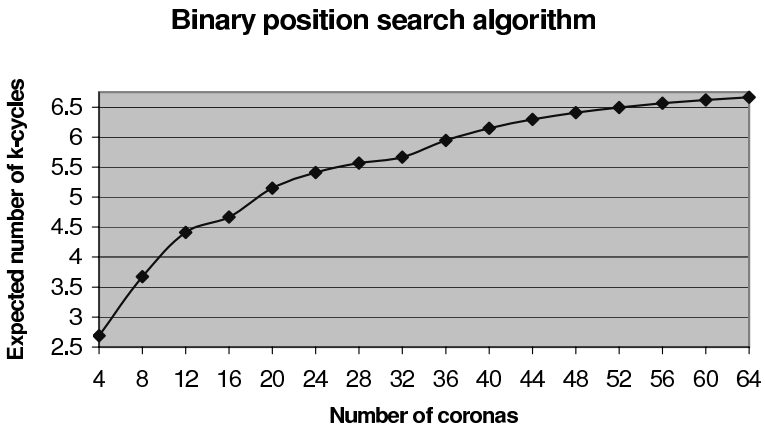

Fig. 11. Expected awake time slots.

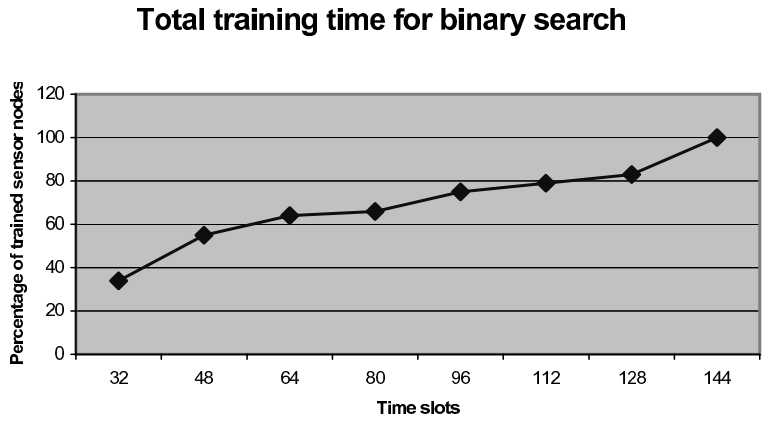

Fig. 12. Total training time for binary search.

shown in Fig. 12 is 144.

The difference may occurs depend on the number of sensor nodes deployed in the interest area and the number of corona being established during the training period. We prefer to use 64 coronas for < 500 sensor nodes.

In Fig. 13 we plotted the difference values of $E[T]$ in theoretical and simulation results. Our simulation results shown less number of cycles needed compare to theoretical results. The difference number of cycles shown is in the range of $[0.1, 0.6]$.

6. Concluding remarks

In this work we have proposed a localization algorithm by training the sensors to impose a coordinate system onto the network. The algorithm is simple and self-organization to train the sensors to learn their coarse-grain location. This algorithm which hybrids the synchronization and training procedure to reduce the sink-cycle that has to be transmitted by the sink for 64 coronas. From these results, multiple-training can be implemented to this protocol which involves more than one sinks establish the

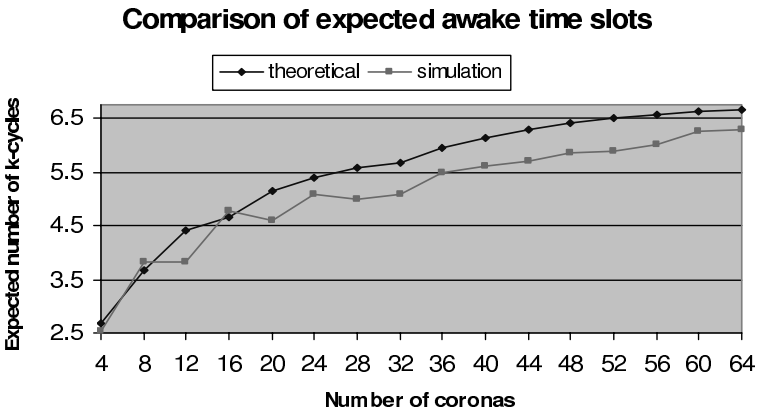

Fig. 13. Comparison of expected awake time slots.

same protocol. For this reason, the results of total training time can be varied and to be an exciting area for further work.

References

[1] J. Agre and L. Clare, An integrated architecture for cooperative sensing networks, *IEEE Computer* **33**(5) (2000), 106–108.
[2] I.F. Akyildiz, W. Su, Y. Sankarasubramanian and E. Cayirci, Wireless sensor networks: A survey, *Computer Networks* **38**(4) (2002), 393–422.
[3] J. Albowicz, A. Chen and L. Zhang, *Recursive Position Estimation in Sensor Networks*, UCLA Internet Research Laboratory
[4] I. Chatzigiannakis and S. Nikoletseas, *A Sleep-Awake Protocol for Information Propagation in Smart Dust Networks*, Proceedings of IEEE International Parallel and Distributed Processing Symposium, Nice, France, April, 2003.
[5] D. Culler, D. Estrin and M. Srivastava, Overview of sensor networks, *IEEE Computer* **37**(8) (2004), 41–49.
[6] P. Enge and P. Misra, Special issue on GPS: the global positioning system, *Proc IEEE* **87** (1999), 3–15.
[7] M. Ilyas and I. Mahgoub, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, CRC Press, 2005, 21(1)–21(2).
[8] Y. Ko and N. Vaidya, *Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms*, IEEE Workshop Mobile Computing Syst. Applications (WMCSA), 1999, 101–110.
[9] Y. Ko and N. Vaidya, *GeoTORA: A Protocol for Geocasting in Mobile Ad Hoc Networks*, 8th Int. Conf. Network Protocols (ICNP), 2000, 240–250.
[10] W.-H. Liao, Y.-C. Tseng, K.L. Lo and J.-P. Sheu, GeoGRID: a geocasting protocol for mobile ad hoc networks based on GRID, *J Internet Technol* **1**(2) (2000), 23–32.
[11] Q. Xu, R. Ishak, S. Olariu and S. Salleh, *On Asynchronous Training in Wireless Sensor Networks*, The Third International Conference on advances in Mobile Multimedia, September, 2005.
[12] Y. Yu, R. Govindan and D. Estrin, Geographical and Energy Aware Routing: A recursive Data Dissemination Protocol for Wireless Sensor Networks, UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001.
[13] A. Wadaa, S. Olariu, L. Wilson, K. Jones and M. Eltoweissy, *Training a Sensor Networks*, MONET. January 2005.
[14] A. Wadaa, S. Olariu, L. Wilson, K. Jones and Q. Xu, *On Training Wireless Sensor Networks, Proc. 3rd International Workshop on Wireless, Mobile and Ad Hoc Networks* (*WMAN'03*). April. Nice, France. 2003.

Ruzana Ishak is a Lecturer of Science Department at Univ. Teknologi Malaysia CityCampus, Kuala Lumpur. She received the B.Sc degree in Computational Maths and CAGD from Univ. Sains Malaysia, Penang in 1993 and the M.Sc degree in Mathematics from Univ.Teknologi Malaysia, Johor Bahru in 2002. She received her Phd in 2007 from the same university on Multi-training techniques for localization in wireless sensor networks. Her research interests are in the area of graph theory application models, wireless sensor network location systems and mobile computing. E-mail: ruzanaishak@yahoo.com.

Stephan Olariu is a tenured full professor in Computer Science at Old Dominion University. He is a world-renowned technologist in the areas of parallel and distributed systems, parallel and distributed architectures and networks. He was invited and visited more than 120 universities and research institutes around the world lecturing on topics ranging from parallel algorithms, to graph theory, to wireless networks and mobile computing, to biology-inspired algorithms and applications, to telemedicine, to wireless location systems, and sensor network applications. Professor Olariu is the Director of the Sensor Networks Research Group at Old Dominion University. He has published 200+ archival journal articles and 100+ conference papers. Professor Olariu earned his BSc, MSc and Ph.D. (Computer Science) at the McGill University, Montreal, Canada. He has received an NSF Research Initiation award. Stephan is an Associate Editor of "Networks" And and serves on the editorial board of "IEEE Transactions on Parallel and Distributed Systems" and "Journal of Parallel and Distributed Computing". Prof. Olariu's current research interests are in the area of parallel and distributed systems, wireless networks performance evaluation and security. E-mail: olariu@cs.odu.edu.

Shaharuddin Salleh is Professor in Computational Mathematics at the Department of Mathematics, Universiti Teknologi Malaysia. He obtained PHD degree in 1998 from the same university. Prof Salleh has published 4 books and 58 technical papers. His research interests include numerical modeling and simulation, parallel algorithms, graph theoretical applications and mobile computing. E-mail: ss@mel.fs.utm.my.