A Geometric Approach to Slot Alignment in Wireless Sensor Networks

Niky Riga Ibrahim Matta Azer Bestavros
Computer Science
Boston University
Email: {inki, matta, best}@cs.bu.edu

***Abstract*—**
Traditionally, slotted communication protocols have employed guard times to delineate and align slots. These guard times may expand the slot duration significantly, especially when clocks are allowed to drift for longer time to reduce clock synchronization overhead. Recently, a new class of lightweight protocols for statistical estimation in wireless sensor networks have been proposed. This new class requires very short transmission durations (jam signals), thus the traditional approach of using guard times would impose significant overhead. We propose a new, more efficient algorithm to align slots. Based on geometrical properties of space, we prove that our approach bounds the slot duration by only a constant factor of what is needed. Furthermore, we show by simulation that this bound is loose and an even smaller slot duration is required, making our approach even more efficient.

I. Introduction

Motivation: Over the past few years, sensor networks have received much attention as they are envisioned to support a wide range of important applications, *e.g.* surveillance systems, biological monitoring systems, environment control systems, equipment supervision systems, etc. A large number of such sensor applications are based on small, inexpensive, battery operated, electronic microsensor devices (*e.g.* Berkeley/Crossbow Motes [1], MIT μAMPS nodes [2]) with radio, sensing and processing components. Due to the size and cost restrictions, these *wireless* sensor devices have limited storage and computation capabilities. Furthermore, access to the sensors may be difficult, or even impossible, after their initial deployment, which implies that the energy expended must be minimized to increase the lifetime of the system.

In order to minimize energy expenditure in such systems, sensors are periodically allowed to shut down their radio and CPU to save energy, which led many researchers to propose slotted protocols [3]–[8]. The nodes in a Wireless Sensor Network (WSN), as in any other distributed system, do not have synchronized clocks. However, the nature of the applications built on top of WSNs, requires some kind of synchronization between the nodes [9], for example, a node should send in a slot during which a neighboring node is scheduled to be awake. How tight or loose, global or local the synchronization is, varies from application to application. In most slotted protocols for WSNs, the nodes run some kind of local synchronization protocol. Slot duration is expanded using guard times in the beginning and end of each slot to ensure alignment. For most protocols, such an expansion is just a small overhead. However, in lightweight protocols, such as those in which nodes transmit for very short durations (jam signals) for the sole purpose of statistical estimation [7], [8], and where the protocol needs to run over multiple contiguous slots, such an expansion might be an overkill. The authors in [7] compute that the necessary slot duration for their protocol is 18μs for an IEEE 802.11a compliant radio.[1] If a more suitable radio for wireless sensors, such as the CC1100, is considered, the same calculation yields a slot size of 39.5μs. At a maximum clock drift of $40ppm$ and, assuming a clock synchronization protocol that runs every 100s, we would require a total guard time of $2 \times 4ms$, which results in a slot duration 200 times larger than necessary.[2]

Our Contributions: We summarize our contributions as follows:

- We propose a new slot alignment algorithm (Section V) that requires a slot expansion of only constant factor of the needed slot duration. The key idea of our algorithm, is having the sensors synchronize their clocks with one of their neighbors before executing the actual slotted protocol. Given the distributed nature of a WSN and the need to minimize message exchange, neighboring nodes might end up synchronizing with different nodes. Using geometrical properties of space, we bound the number of such independent alignments to 22. This constant factor of 22 is in contrast to the traditional slot alignment based on guard times (Section IV), which may require a much longer slot duration that is dependent on drifts in the clocks. Furthermore, our approach uses only short transmissions to achieve slot alignment, thus minimizing the energy expenditure overhead.
- Through simulations (Section VI), we show that the theoretic constant-factor bound on slot duration, is in fact too loose for the average case scenario and a much smaller value can be used in practice, thus making our approach even more efficient.

[1]This minimum slot duration is computed as the sum: $slot\_time = max(t_{rx/tx}, t_{tx/rx}) + 2 \times t_{channel\_delay} + t_{carrier\_sense}$, where $t_{rx/tx}$ and $t_{tx/rx}$ is the time the radio needs to switch from receive mode to transmit and vice versa.

[2]Given that the clock might drift at most $40\mu s$ in every $1s$, after $100s$ the maximum clock drift is $40 \times 10^{-6} * 100 = 4ms$. This means that two clocks that started synchronized might have a maximum difference of $2 \times 4ms$; the clock drift might be either positive or negative.

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

II. Problem Statement

For the rest of the paper, we assume that sensors run a generic slotted protocol. Time is divided into slots, and a node can transmit in any of the slots. Whether the slot-based protocol is used for communication, e.g. [3]–[6], or for statistical estimation [7], [8], it is essential that no single transmission spans two slots in order to either avoid collisions or obtain accurate statistics.

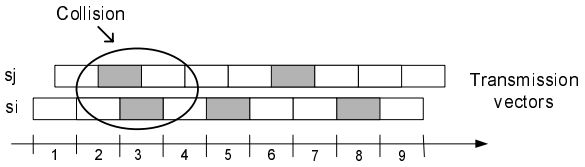

Fig. 1. Effect of clock skewness on a slotted protocol.

Figure 1 depicts the effect of clock skewness on a slotted protocol. In this example, the slot duration is equal to the transmission time. It is obvious how the misalignment of slots can induce collisions.

Definition 1 (Slot alignment): A network achieves *slot alignment* when no single transmission spans two slots for any of the receiving nodes.

Figure 2 shows an example of a network that achieves slot alignment. The dotted vertical lines denote the slot boundaries according to sensor s_i. In this case, every transmission (shown as shaded areas) falls within the boundaries of a slot.

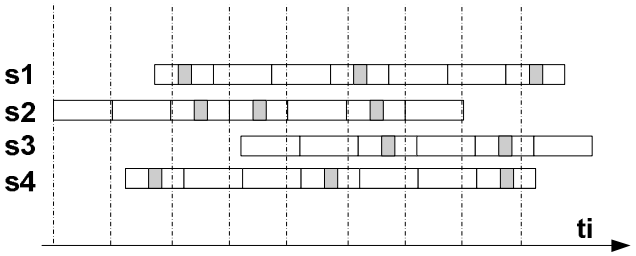

Fig. 2. Example of slot alignment. The timeline is with respect to sensor s_i. The dotted vertical lines denote the slot boundaries for sensor s_i.

In this paper, we present an efficient algorithm to achieve network-wide slot alignment.

III. Definitions

Let d_{tx} denote the duration of one transmission, and d_{slot} denote the duration of a slot. Let $S = \{s_1, ..., s_N\}$, be the set of all the sensor nodes in the network, where N is the network size.

Definition 2 (Neighborhood S_i): The neighborhood S_i of sensor node s_i is the set of nodes that node s_i can directly communicate with.

Definition 3 (Clock Skew δ_{ij}): The clock skew between two nodes i and j, denoted by δ_{ij}, is defined to be the difference between the clocks of the two sensors. If t_i is the current time according to sensor s_i, and t_j according to sensor s_j, then

$$\delta_{ij} = |t_i - t_j| \quad (1)$$

Henceforth, we assume that any kind of WSN requires that the nodes be at least locally synchronized. However, there is no assumption on how loose or tight, this synchronization must be. There have been significant work in the area of clock synchronization for wireless sensor networks [9]–[11]. Each one of these approaches provides an upper bound δ on the clock skew which is a parameter of the system.

Definition 4 (Maximum Local Clock Skew δ): Let δ denote the maximum local clock skew. $\forall$ neighborhoods S_i, $\delta_{ij} \leq \delta$, $\forall s_j \in S_i$, at any point in time.

For the remainder of the paper we assume that the time is measured in multiples of the system's clock tick.

IV. Slot Alignment using Guard Times

In this section, we present a straightforward way to achieve slot alignment, given δ. Let $d_{slot} = d_{tx} + 2\delta$. In each slot, a node waits for δ time period before it transmits. For any sensor s_i, this ensures that its slot has started when nodes with more advanced clocks might transmit, and that the slot is extended enough to include transmissions of nodes with slower clocks.

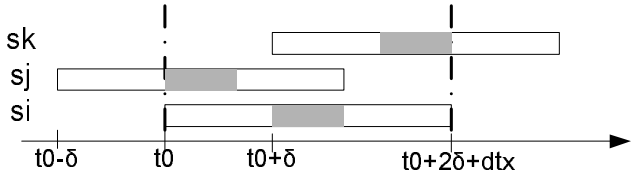

Fig. 3. This is how the slots of sensors s_j and s_k are aligned with sensor s_i. The timeline in the bottom of the figure represents the time according to sensor s_i.

Let's consider Figure 3. Let $s_j, s_k \in S_i$ and let $\delta_{ij} = \delta_{ik} = \delta$. In this example, the clock of s_j is faster than s_i, while the clock of s_k is slower; s_j and s_k are not within communication range. The shaded area within each slot is the transmission period. The slot for s_i starts at time t_0. For s_j, the slot has started δ before t_0. At t_0, the transmission period for s_j starts, while at time $t_0 + 2\delta + d_{tx}$, the transmission period for node s_k ends. The transmission period of any other sensor that has a clock skew less than δ starts after the transmission of s_j, and finishes before the transmission of s_k, so all the transmissions of nodes from the neighborhood of s_i start and finish within a slot. The generalization for the rest of the slots is straightforward. Using this method the network achieves slot alignment.

The overhead, H_{guard}, introduced by the above approach is measured in terms of the extra time introduced in the running time due to slot alignment. Let n be the total number of slots. Let $\delta = c \times d_{tx}$ for $c \geq 1$. Under perfect synchronization, *i.e.* $c = 0$, the running time is $T_{min} = n \times d_{tx}$, while using the

1930-529X/07/$25.00 © 2007 IEEE

guard-time approach for $c > 0$, the running time is

$$\begin{aligned} T_{guard} &= n \times (d_{tx} + 2\delta) \\ &= n \times (d_{tx} + 2 \times c \times d_{tx}) \end{aligned}$$

Thus the overhead $H_{guard} = T_{guard} - T_{min} = 2n \times c \times d_{tx}$.

V. Geometric Approach to Slot Alignment

If $\delta \gg d_{tx}$ then the overhead introduced by the above approach is significant. To reduce this overhead, we propose a new method for slot alignment among nodes. In our approach, nodes first try to locally synchronize their clocks. To this end, our approach uses only one transmission per node. The slot duration is expanded enough to accommodate different schedules from different nodes.

Let the duration of each slot be $d_{slot} = \alpha \times d_{tx}$, where α is a network-wide constant. We compute a bound on the value of α later in this section. Each slot is divided into α minislots of d_{tx} duration each. A node can transmit in only one minislot called the *transmission* minislot. In order to achieve slot alignment, the transmission minislot should not span two different slots of a neighbor.

The Geometric Slot Alignment (GSA) algorithm (Algorithm 1) achieves slot alignment among the nodes of a wireless network. After all the nodes have executed this algorithm, each node:

- has aligned its minislots, *i.e.* it has computed the start time of each minislot,
- has computed its transmission minislots, and
- has determined the boundaries of each slot, *i.e.* which minislots are the first minislot of each slot.

Algorithm 1 $GeometricSlotAlignment(\delta, d_{tx}, d_{slot})$

Input: δ the maximum local clock skew, d_{tx} the duration of one minislot, d_{slot} the duration of one slot

1: $t_{tx} = -1$;
2: $t_{start} = t_{now} = get\_local\_time()$;
3: **while** $t_{now} \leq t_{start} + \delta$ **do**
4: $t_{now} = get\_local\_time()$;
5: **if** $(channel\_is\_busy == TRUE)$ **then**
6: $t_{tx} = t_{start} + \delta + [d_{slot} - (t_{start} + \delta - t_{now}) \% d_{slot}]$;
7: break;
8: **end if**
9: **end while**
10: **if** $t_{tx} == -1$ **then**
11: $t_{tx} = get\_local\_time()$;
12: **end if**
13: $transmit(t_{tx})$; //Transmit at time t_{tx} for one minislot
14: $t_{first\_minislot} = monitor\_channel(t_{start} + 2 \times \delta + d_{slot} + d_{tx} - get\_local\_time())$

The function that is called in line 14 of the algorithm, monitors the channel for the duration that is provided as input, and returns the time that the first slot starts, *i.e.* the time when the first minislot of the first slot starts. The first minislot is chosen carefully to ensure that the transmission minislots of neighboring nodes are fully contained within only one slot. As we prove later, if α is large enough, such minislot always exists. Consider the case of Figure 4. In this example, node s_i has only 4 neighbors and $\alpha = 9$. As we prove later in this section, when s_i finishes Algorithm 1, it has heard one transmission from every neighbor. Let's say that the nodes have aligned as shown in Figure 4. In this case, node s_i will choose minislot number 5 as the start of the slot, since this minislot is the first whose *beginning* is idle, and as we show later, this guarantees that no transmission will span two slots of node s_i. The *transmission* minislot is chosen at the end of the algorithm based on the value of t_{tx}, the first transmission time assigned by Algorithm 1, and the fact that the transmission minislot is periodic and is repeated every α minislots. In this example, $t_{tx} = 1$.

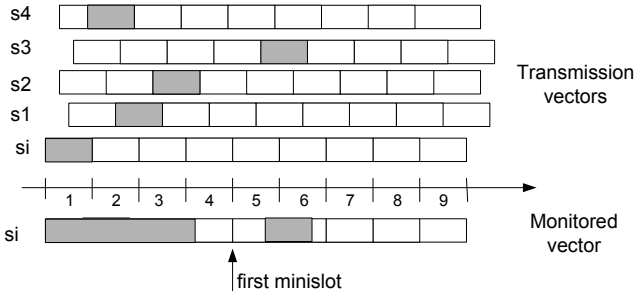

Fig. 4. An example of how node s_i is choosing its first minislot, after executing Algorithm 1. $\alpha = 9$, and s_i has 4 neighbors. In this example, s_i will choose the minislot number 5 as its first minislot.

Claim 1: After all the nodes have executed Algorithm 1, and for $\alpha \geq 23$, the network is slot aligned.

In order to prove the above claim, we first prove some other helpful claims. Only for the sake of analysis, we assume that there is a global clock according to which we timestamp the events. Let t^i_{start} denote the time at which node s_i starts executing the algorithm. Given that the maximum local clock skew in the network is δ, we know that

$$\forall s_i \in S,\ |t^i_{start} - t^j_{start}| \leq \delta,\ \forall s_j \in S_i \qquad (2)$$

According to Algorithm 1, every node s_i will either choose its schedule independent of everyone else (line 11), or it will synchronize its transmission minislots according to the first neighbor that transmits (line 6). Let t^i_{tx}, be the time that node s_i transmits. Since the nodes synchronize their schedules based on their first transmission, it is essential that $t^i_{tx} \geq t^j_{start}, \forall s_j \in S_i$.

Claim 2: For all nodes s_i, $t^i_{tx} \geq t^j_{start},\ \forall s_j \in S_i$.

Proof: The transmission time is set in either line 6 or line 11. If node s_i chooses its schedule independent of everyone else (line 11), the first transmission starts exactly at time $t^i_{start} + \delta$, which ensures that all nodes in S_i have started executing Algorithm 1. If node s_i synchronizes its schedule

1930-529X/07/$25.00 © 2007 IEEE

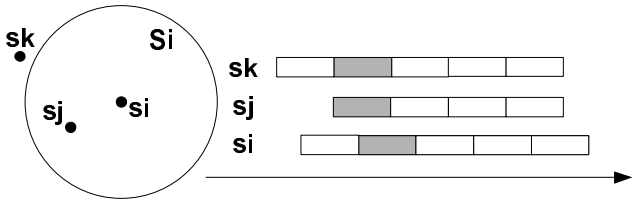

Fig. 5. The schedule of the minislots for nodes s_i, s_j and s_k are shown. Nodes s_k and s_i are not neighbors. The shaded minislots denote the transmission minislots of each node.

based on the transmission of another node s_k, then the first transmission is scheduled in line 6 of the algorithm. Obviously, the term $[d_{slot} - (t^i_{start} + \delta - t_{now})\%d_{slot}]$ is positive and thus $t^i_{tx} \geq t^i_{start} + \delta \geq t^j_{start}, \forall s_j \in S_i$. ■

In order for any node s_i to be able to determine the slot boundaries, all the nodes in its neighborhood should transmit once denoting the alignment of their minislots.

Claim 3: For every node s_i, at time $t^i_{start} + 2 \times \delta + d_{slot} + d_{tx}$, all nodes in S_i have transmitted at least once.

Proof: According to claim 2, for every node s_j, it is true that $t^j_{tx} \geq t^j_{start} + \delta \geq t^i_{start}$. Moreover, since $d_{slot} - (t^j_{start} + \delta - t_{now})\%d_{slot} \leq d_{slot} < d_{slot} + d_{tx}$, then from Algorithm 1, $t^j_{tx} \leq t^j_{start} + \delta + d_{slot} + d_{tx}$. According to Equation (2), $t^j_{start} - t^i_{start} \leq t^j_{start} - min_{k,s_k \in S_j} t^k_{start} \leq \delta \Leftrightarrow t^j_{start} \leq t^i_{start} + \delta$. Combining all the above equations, we get:

$$t^i_{start} \leq t^j_{tx} \leq t^i_{start} + 2 \times \delta + d_{slot} + d_{tx}$$

■

Based on Algorithm 1, at most one node s_j in every neighborhood S_i can choose its schedule independently. The rest of the nodes in this neighborhood will either synchronize with this node or with another node s_k of their neighborhood such that $s_k \notin S_i$.

Let's assume that node s_i chooses its schedule independently, and another node s_j has already been synchronized with node $s_k \notin S_i$. Since node $s_k \notin S_i$ and $s_i \notin S_k$, we say these two schedules are *independent*. In the worse case, the minislots' schedules of nodes s_i and s_j are not aligned and the two transmission minislots can not be separated into different slots—see Figure 5. In this case, $d_{slot} \geq 2 \times d_{tx}$ ensures that no transmission minislot spans two slots of another neighboring node.

If in S_i, there are α independent schedules, then $d_{slot} \geq \alpha \times d_{tx}$ ensures that the nodes can achieve loose slot synchronization. If the maximum number of independent schedules in a neighborhood is computed, a lower bound on the slot size can also be computed.

Claim 4: In any neighborhood S_i there can be at most 22 independent schedules.

Before proving this claim (restated in claim 7 below), we first prove other helpful claims.

Every node in S_i, will synchronize with the schedule of a node in the two-hop neighborhood of s_i. Hence, only nodes in the two-hop neighborhood, *i.e.* nodes within a radius of $2r$ from s_i—r denotes the communication range—can affect the schedule of nodes in S_i.[3] As mentioned earlier, only one node per neighborhood can choose its schedule independently, which means that if two nodes chose their schedule independently, the distance between them is at least r, *i.e.* they are out of communication range. Based on this observation our problem becomes that of computing the maximum number of nodes in a two-hop neighborhood of any node so that the pairwise distance between any two of them is at least r. More formally we want to prove the following:

Claim 5: In any circle of radius $2r$ there can be at most 22 points, so that the pairwise distance between any two of them is at least r.

In order to prove the above claim, we first prove the following:

Claim 6: In a circle of radius R, we can fit n points so that the pairwise distance is at least d, if and only if we can pack n non-intersecting circles with radius $\frac{d}{2}$ inside the circle of radius $R + \frac{d}{2}$.

$\Rightarrow$ If n points with minimum pairwise distance d can fit in a circle of radius R, then n non-intersecting circles can be packed in a circle of radius $R + \frac{d}{2}$.

Proof: Place the n points within the circle of radius R, and with each of these points as a center draw a circle of radius $\frac{d}{2}$. These n circles are non-intersecting since the minimum distance between the centers is d. It is straightforward that these n circles will be contained in a circle of radius $R + \frac{d}{2}$. ■

$\Leftarrow$ If n non-intersecting circles can be packed in a circle of radius $R + \frac{d}{2}$, then n points with minimum pairwise distance d can fit in a circle of radius R.

Proof: If we pack n non-intersecting circles of radius $\frac{d}{2}$ in a circle of radius $R + \frac{d}{2}$, then we also have n points, the centers of the circles, that are at distance at least d from each other. It is straightforward to see that the centers of all the n circles are at a distance of at most R from the center of the circle with radius $R + \frac{d}{2}$. ■

Based on claim 6 we restate claim 4 as follows:

Claim 7: In a circle of radius $2r + \frac{r}{2}$, we can pack at most 22 non-intersecting circles of radius $\frac{r}{2}$.

Proof: The problem of packing circles in a 2-d plane is a well-studied problem. Fejes Toth has proved that the maximum density any circle packing can achieve in a 2-dimensional plane, is $\frac{\pi}{\sqrt{12}}$ [12]. Let n be the maximum number of circles of radius $\frac{r}{2}$ that can be packed in a circle of radius $2r + \frac{r}{2}$. We know that

$$n \times \pi(\frac{r}{2})^2 / \pi(\frac{5r}{2})^2 \leq \frac{\pi}{\sqrt{12}} \Leftrightarrow \frac{n}{25} \leq \frac{\pi}{\sqrt{12}} \Leftrightarrow n \leq 22.66$$

■

Now, we can prove claim 1.

[3]For analytical simplicity, we assume that the communication range is the same for all nodes in the network and equal to r. The analysis can be extended to handle the case of heterogeneous, asymmetric communication ranges.

1930-529X/07/$25.00 © 2007 IEEE

Proof: Given that we can have at most 22 independent schedules, every node will hear at most 22 transmissions in each slot. If the duration of each slot is equivalent to 23 minislots, *i.e.* $\alpha = 23$, there should be at least one minislot for every node such that there is no transmission in the beginning of that minislot. Based on claim 3, we know that a node has heard a transmission from all its neighbors before the end of Algorithm 1. We also know that the transmission minislots are periodic with a period of α minislots. Given this information, every node can compute the transmission minislots of all its neighbors for a period of α minislots and find the minislot during which no one is transmitting in its beginning and make this the first minislot of each slot—which is what the function in line 14 of Algorithm 1 does. In this way, every node manages to delineate the transmission minislots of its neighbors and achieve slot alignment. ■

Going back to the example in Figure 4, sensor s_i picks minislot 5 as the start of the slot, its first transmission minislot becomes $1+9 = 10$, and its subsequent transmission minislots would repeat every 9 minislots.

Algorithm 1 should be executed before the actual slotted protocol. In order to compute the overhead of our approach, we assume that the slotted protocol needs to run for n slots.

The execution time of Algorithm 1 is $2 \times \delta + d_{slot} + d_{tx}$.

Thus, the total time required is

$$\begin{aligned} T_{gsa} &= 2 \times \delta + d_{slot} + d_{tx} + n \times d_{slot} \\ &= 2 \times c \times d_{tx} + (\alpha + 1) \times d_{tx} + n \times \alpha d_{tx} \\ &= (2c + \alpha + 1 + \alpha n) \times d_{tx} \end{aligned}$$

The minimum running time, under perfect no-overhead synchronization, is $T_{min} = n \times d_{tx}$.

The overhead, H_{gsa}, is thus given by $T_{gsa} - T_{min}$:

$$H_{gsa} = (2c + 24) \times d_{tx} + (\alpha - 1)n \times d_{tx}$$

Now we compute the critical value for c which determines which type of slot alignment is more efficient to use. Our approach is more efficient if the following is true:

$$\begin{aligned} H_{gsa} &< H_{guard} \\ c &> \frac{\alpha + 1 + (\alpha - 1)n}{2n - 2} \end{aligned}$$

VI. Performance Evaluation

In order to experimentally evaluate the GSA algorithm, we implemented and tested the algorithm in matlab [13]. The first set of experiments we ran was for the purpose of testing the effect of the parameter α. We expect that as the density of the network increases, the minimum α required to achieve slot alignment will also increase up to the value of 23. In a field of $100m \times 100m$, we created random topologies. We increased the total number of nodes in the network while keeping the communication radius steady at $10m$. For this simulation, we used $\delta = 4ms$ and $d_{tx} = 40\mu s$.

Figure 6 shows the average number of nodes that failed to align their minislots—*i.e.* after the execution of our GSA algorithm, they could not choose a minislot to separate all the transmission minislots in separate slots—for increasing values of α. Each line represents a different network size. The results shown are the average of 20 runs along with 95% confidence intervals. From this figure, it is obvious that the minimum α for which all the nodes achieved slot alignment—a value of 9—is well below the theoretical value of 23, even for extremely dense networks. The neighborhood sizes for each network are presented in Table I.

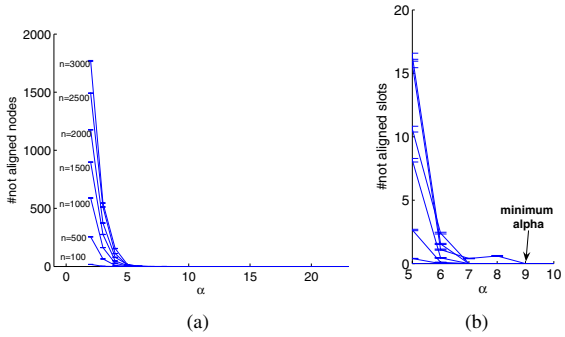

Fig. 6. (a) Average number of nodes that failed to align for increasing value of α and for varying network sizes; (b) a zoomed in version of (a) to show detail.

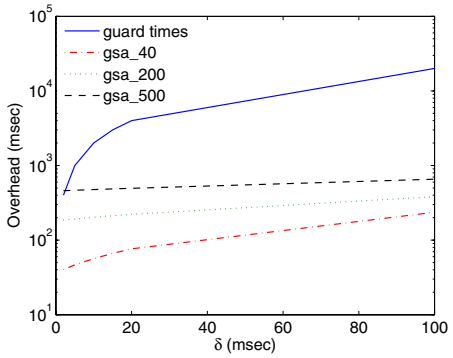

Fig. 7. The time overhead in the protocol's execution for different values of δ. The protocol needs to run for n=100 slots. The different plots for the GSA algorithm correspond to varying d_{tx} for the protocol (40, 200 and 500μs) with $\alpha = 10$.

TABLE I
AVERAGE NEIGHBORHOOD SIZES OVER 20 NETWORKS

Net_Size	mean(Neigh_Size)	min(Neigh_Size)	max(Neigh_Size)
100	3.8	1.0	7.9
500	15.4	3.9	27.2
1000	29.7	8.0	48.3
1500	44.1	13.0	67.7
2000	58.6	16.3	87.8
2500	73.1	20.2	106.2
3000	87.5	23.8	125.5

Figure 7 shows the time overhead in the execution time of a protocol that needs to run for 100 slots, for varying δ. We observe that the overhead when using guard times grows really fast (note that the y-axis is in logarithmic scale). The different

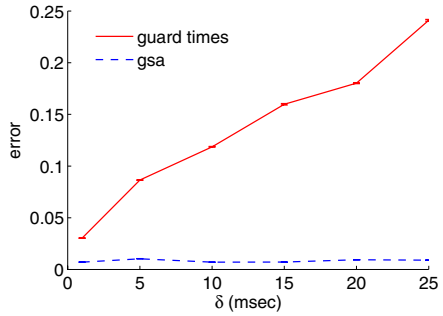

Fig. 8. The average error in computing the local density using DIP, when guard times and gsa are used for slot alignment for varying δ. The total running time of DIP is constant at $1s$, d_{tx} is set to $40\mu s$ and $\alpha = 10$.

GSA plots correspond to different d_{tx} values of $40, 200$ and $500\mu s$, with α set to 10—the value of α was set empirically based on the results of Figure 6. For the guard-times approach, $d_{tx} = 40\mu s$.

Next, we wanted to evaluate the effect of the overhead on a protocol with small d_{tx}. We used DIP [8], a statistical protocol that estimates the local density for each node. We kept the total execution time of DIP constant, and we varied the value of δ. Figure 8 shows the average error in DIP's estimation of the local density over 10 runs along with 95% confidence intervals. We observe that the low overhead of our GSA-based slot alignment affords DIP enough time to collect much more reliable statistics, compared to the traditional approach of using guard times.

VII. Related Work and Conclusion

Prior research on slot synchronization in WSNs has focused on reducing the "guard times" around a slot. However, this reduction in overhead usually comes at the expense of running the clock synchronization protocol more frequently to reduce clock drifts. Local synchronization protocols require extra coordination to achieve network-wide slot alignment [14], and may be slow to converge [15]. On the other hand, network-wide approaches require building spanning trees to achieve clock synchronization [16]. These latter approaches vary in their tradeoff between the speed and the optimality of tree construction [17]—a rapid tree construction may sacrifice delay and efficiency in propagating synchronization traffic.

Motivated by lightweight slotted protocols, such as those in which nodes transmit for very short durations (jam signals) for the sole purpose of statistical estimation [7], [8], we take a radically different approach to slot alignment. We show that a slot duration of a (practically small) constant-factor of the (short) transmission time suffices for nodes to align their slots by overhearing other nodes, so no transmission spans more than one slot.

Acknowledgment

This work was supported in part by NSF CNS Cybertrust Award 0524477, CNS ITR Award 0205294, and EIA RI Award 0202067.

References

[1] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for Smart Dust," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM Press, 1999, pp. 271–278.

[2] A. Chandrakasan, R. Min, M. Bharwaj, S.-H. Cho, and A. Wang, "Power Aware Wireless Microsensor Systems," in *Proceedings of the 28th European Solid-State Circuits Conference (ESSCIRC)*, September 2002.

[3] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a Hybrid MAC for Wireless Sensor Networks," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05)*. New York, NY, USA: ACM Press, 2005, pp. 90–101. [Online]. Available: http://portal.acm.org/citation.cfm?id=1098929

[4] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*. New York, NY, USA: ACM Press, 2004, pp. 95–107. [Online]. Available: http://portal.acm.org/citation.cfm?id=1031508

[5] R. Rozovsky and P. R. Kumar, "SEEDEX: a MAC Protocol for Ad Hoc Networks," in *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01)*. New York, NY, USA: ACM Press, 2001, pp. 67–75. [Online]. Available: http://dx.doi.org/10.1145/501426.501427

[6] J. R. et al., "Joint Architecture Vision for Low Energy Networking (JAVeLEN)–DARPA-ATO Connectionless Networking Program," 2004, BBN Technichal Report. [Online]. Available: www.jasonredi.com

[7] A. Krohn, M. Beigl, and S. Wendhack, "SDJS: Efficient Statistics in Wireless Networks," in *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 262–270.

[8] N. Riga, I. Matta, and A. Bestavros, "DIP: Density Inference Protocol for wireless sensor networks and its application to density-unbiased statistics," in *Proceedings of the Second International Workshop on Sensor and Actor Network Protocols and Applications (SANPA)*, 2004.

[9] J. Elson, L. Girod, and D. Estrin, "Fine-grained Network Time Synchronization using Reference Broadcasts," *Special Interest Group on Operating Systems (SIGOPS) Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.

[10] J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks," in *Proceedings of the 15th International Parallel & Distributed Processing Symposium (IPDPS)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 186.

[11] M. Sichitiu and C. Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2003)*, 2003.

[12] F. Toth, *Regular Figures*. New York: Pergamon, Macmillan, 1964.

[13] I. The MathWorks, "MATLAB," 2000, http://www.mathworks.com/.

[14] T. Salonidis and L. Tassiulas, "Performance Issues of Bluetooth Scatternets and other Asynchronous TDMA Ad Hoc Networks," in *Proceedings of MoMuC*, 2003.

[15] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," in *Proceedings of IEEE Infocom*, 2004.

[16] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing Sync Protocol for Sensor Networks," in *Proceedings of ACM SenSys*, 2003.

[17] L. Dai, P. Basu, and J. Redi, "An Energy Efficient and Accurate Slot Synchronization Scheme for Wireless Sensor Networks," in *Proceedings of BROADNETS*, 2006.