Temporal Partition in Sensor Networks

Ted Herman[1], Sriram Pemmaraju[1], Laurence Pilard[1,2], and Morten Mjelde[1,3]

[1] Department of Computer Science,
University of Iowa, Iowa City, IA 52242-1419, U.S.A.
[herman,sriram]@cs.uiowa.edu
[2] Somewhere in France pilard.laurence@gmail.com
[3] Department of Informatics
University of Bergen
N-5020, Bergen, Norway mortenm@ii.uib.no

Abstract. Sensor networks are composed of nodes embedded in physical environments where applications may be tasked to run for years without human maintenance and without continuous external power supply. Strategies for power conservation are thus important in sensor network protocols and system architecture. One such strategy is to arrange node sleeping schedules so that radios are powered off until communication is necessary. Nodes cannot receive messages during periods when the radio is turned off. In this setting, there can arise situations where groups of network nodes have somehow become *temporally partitioned*: due to having different sleeping schedules, groups of nodes could be unaware of each other. The paper presents several self-stabilizing protocols to solve the problem of temporal partition; starting from an arbitrary temporally partitioned state, these protocols lead the network to a state in which all nodes have a perfectly aligned sleep schedule. Our techniques include using randomly chosen relatively prime sleep periods and occasional, and possibly random, probing of extra time slots. Our protocols aim for fast convergence while imposing only a small energy consumption overhead.

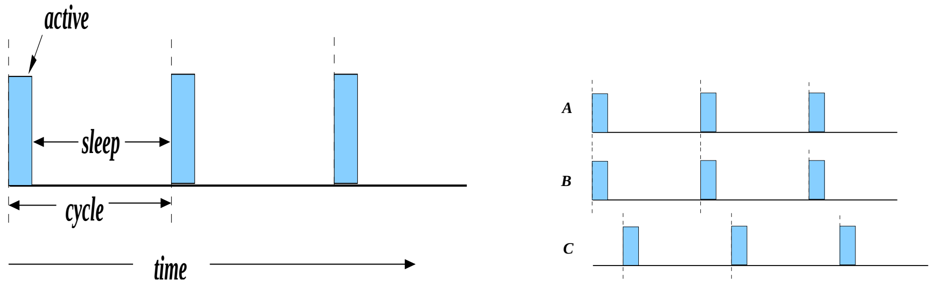
1 Introduction

Efficient power utilization is widely studied in the Wireless Sensor Networks (WSN) community. WSNs are composed of nodes embedded in physical environments. In the literature, WSNs vary widely in their capabilities, ranging from primitive computing devices with a few thousand bytes of memory to cell-phone class machines that have tens of megabytes of memory. Similarly, there can be wide variation of communication bandwidth, computing speed, and sensing capabilities. Network architectures for WSNs may depend on a gateway that is the source of commands and the sink for sensor data; or the architecture may be fully distributed, allowing even for mobile nodes. In some scenarios, selected nodes could be attached to powered infrastructure, whereas others depend on battery power, possibly supplemented by intermittent power supply (e.g., solar or vibrational energy harvesting). Finally, users specify desired latency, network lifetime, sensing accuracy, and so forth, leading to applications with differing structures (sensed values can either be triggered by events and reported quickly or can be sampled periodically and aggregated to be collected later). Taking this broad spectrum of architectures and capabilities into account, efficient power utilization depends on the selection of model details. Section 2 presents the model choices for this paper.

Among the ideas for reducing power consumption, powering down the processor is the simplest to implement (modern processor architectures typically enable fast switching to low power idle states). Auxiliary devices in a sensor node, including flash memory and radio, can also be put into low power modes or turned off completely. To illustrate the utility of turning off the radio, we consider power metrics for a widely used radio chip, the CC2420. During a transmit operation, the radio draws between 8.5 to 17.4 milliAmps, depending on which level of transmission power is selected; during a receive operation, the radio draws 18.8 milliAmps. The radio can also be in *idle mode*, ready to receive a message or sensing radio frequency activity. But, even in idle mode, the radio uses 426 microAmps. Typical message transmission (or reception) time is 1.472 milliseconds. It follows that receiving a message consumes the same power as running in idle mode only for about 65 milliseconds. Thus a significant power saving is gained by turning the radio off rather than leaving it in idle mode. With a battery rated at 1000 milliAmp-hours, maintaining the radio in idle mode consumes the battery supply in a few months, which may be too short for some of the applications being considered for WSNs.

The WSN literature refers to techniques that power off the radio (and other devices) intermittently as duty cycling. A common strategy is to arrange periodic time intervals, called cycles, that begin with the radio on for some fixed period of activity, followed by turning the radio off for the remainder of the cycle. The fixed period with the radio on is called the *active period*, and the remainder of the cycle is the *sleeping period* (see Figure 1(a)). The *duty cycle* is defined as the ratio of the active period length to the length of the entire cycle (sum of active and sleeping periods). Typical WSN targets for duty cycles are 1% or below, which can extend to years the life of a sensor network running on batteries. Though duty cycling seems obviously a good idea to conserve power, there is some risk associated with duty cycle implementation, as the following paragraph explains.

Selection of duty cycle parameters must be balanced against application and network protocol performance. The question of what are optimal parameters for duty cycling has been addressed in previous research (Section 3 covers some results). Some other issues concerning duty cycle implementation are robustness to deployment errors, adaptation dynamic changes to the network, and mobility of sensor nodes. The basic problem we examine in this paper is *temporal partition.* If different subnetworks of a WSN follow duty cycles that are "displaced" such that one subnetwork is sleeping while another is active, can the duty cycling protocol automatically bring the subnetworks together to a common duty cycle? At any time, a set R of nodes in the WSN are called *aligned* if their cycles have the same lengths (active and sleeping periods) and the time instants of when the



(a) (b)

Fig. 1. (a) The typical cycle of an active period followed by a sleep period. (b) Nodes A and B are aligned, whereas node pairs (A, C) and (B, C) are displaced.

next active period starts, for nodes in R, all lie in some time interval of length ϵ (a platform-dependent tolerance factor related to timekeeping abilities). If a pair (x, y) of nodes is not aligned, then we say x and y are *displaced*; more generally, if a set R is aligned and a set S is aligned, but if $R \cup S$ is not aligned, then R and S are displaced (see Figure 1(b)). This terminology enables a succinct statement of the problem we consider, which is a problem of self-stabilization: starting from an initial state where each node has some arbitrary cycle, does the protocol eventually converge to state where the set of all nodes is aligned, and remains aligned thereafter? Implicit in this problem statement is the fact that a pair of displaced nodes may have no awareness of each other and cannot gain this awareness until they are aligned.

One could argue that, in practice, temporal partition rarely if ever occurs, because transient faults of state are very rare (we do not have any data on the occurrence of such faults) and networks are carefully deployed. The addition of new nodes to a network is simple to accommodate: after initialization, new nodes remain active until they overhear the current cycle and then join the network gracefully. Nevertheless, a protocol that can stabilize from temporal partition has the virtue of relaxing constraints on deployment, say, that nodes must be initialized in a controlled order, or disallowing that some group of nodes accidentally initialized together could later be brought to the network as an active group with their own established cycle.

2 Assumptions, Definitions, and Results

If two WSN nodes p and q are in bidirectional communication range, we say there is a link between p and q. Let G be the graph induced by the nodes and bidirectional links; we assume G is connected at any instant. Nodes have immutable, distinct identifiers. A subset S of nodes in G is *stably aligned* if all nodes in S are aligned and remain aligned forever in the absence of any transient faults in the network. The length of the active period is given; we assume that this is a fixed constant and sufficient for a node to perform local bidirectional communication (i.e., with all neighbors). This assumption is partly justified by the *local broadcast model* of communication that is typically used for WSNs; in this model a message transmitted by a node can be heard by *all* other nodes in its communication range. Our protocols remain correct even if the active periods have larger length, e.g., proportional to the diameter of the network, and if these lengths vary from cycle to cycle. However, our running time analysis is considerably simplified by this assumption. To simplify exposition, we normalize the active period length and take it to be 1 time unit. We are also given an upper bound, denoted sleep_{upper}, on the length of the sleeping period. This upper bound is dictated by application latency and periodic reporting specification. Given such an upper bound, $1/(1+\mathsf{sleep}_{upper})$ is a lower bound on the duty cycle that our protocols can achieve.

We can now state the *temporal partition* problem precisely. The nodes of G are partitioned into subsets $N_1, N_2, \ldots$ such that nodes in each subset N_i are initially aligned, whereas nodes in any pair of distinct subsets (N_i, N_j), $i \neq j$, are displaced. Devise a self-stabilizing protocol such that the set of all nodes in G is eventually stably aligned. We use two measures for the performance of our protocols: (i) duty cycle and (ii) convergence time, which we define (as usual) for deterministic protocols to be the worst case time from initial state to a state in which the set of all nodes in G is stably aligned. For randomized protocols we compute the expected convergence time and when possible the convergence time guaranteed with high probability (i.e., $1-1/n$, where n is the number of nodes in the network).

We now state our assumptions related to other important issues such as communication, mobility, and clock synchrony. Nodes send and receive messages to communicate and transmission of a message is not acknowledged. For simplicity of exposition, we assume that messages are not lost. However, our results easily extend to the case where messages can be lost, but if (p, q) remains a bidirectional link for some constant number of cycles, then at least one of p's messages is received by q and vice versa. For simplicity of analysis, we assume that nodes are not mobile, though our protocols are correct even in the presence of limited node mobility. Sensor nodes have discrete clocks that can provide relative timing functions and scheduled wakeup signals and cycle scheduling is enabled by clock synchronization. We assume that a clock synchronization protocol exchanges messages during the active period of the cycle. Consider a network of two nodes, (p, q) that are displaced. The nodes remain displaced at least until some communication occurs between them, so that they can adjust their cycles to become aligned. It is therefore necessary that active periods of the two nodes overlap for a long enough period of time to enable the requisite communication. Whereas clocks are discrete, the amount of displacement between clocks of different nodes may be an arbitrarily small real number. To avoid dealing with fractional overlap between active periods of different nodes, we make the simplifying assumption that the clock "ticks" of all clocks are perfectly aligned. For example, when $clock(x)$ reads 10 units, at that very instant $clock(y)$ reads t time units for some integral t.

Results. The paper presents two classes of self-stabilizing protocols for the temporal partition problem. The first class of protocols (Section 4.1) use a "no cost" approach, which permits nodes to *only* vary their sleep periods and does not allow them to remain active outside their given active periods. This class of protocols provide $O(diam(G) \cdot z^2)$ convergence time with $1/z$ duty cycle for any $z \leq \mathsf{sleep}_{upper}$. The key step in these protocols is a randomized choice of relatively prime sleep periods. The second class of protocols (Section 4.2) use *probing*. These protocols permit nodes to remain active for a small number of time slots outside their active period, for the purposes of probing other components. We consider deterministic as well as randomized probing techniques. We present a family of deterministic probing protocols that provide $O(diam(G) \cdot \frac{z^2}{c})$ convergence time with a duty cycle of $\frac{1+c}{z}$, for any integer c, $1 \leq c \leq z-1$. We show that these protocols are optimal by providing matching lower bounds. For dense, constant-diameter networks, randomized probing provides $O(z \cdot (\log z + \log \log n))$ convergence time with $2/z$ duty cycle, for any $z \leq \mathsf{sleep}_{upper}$. We provide a comparison of the results for these classes of protocols in Section 5.

3 Related Work

In WSN research, problems of power conservation are found at all layers of system architecture. Low-power radio operation has been investigated primarily as part of the MAC layer or at the application layer. The MAC layer encapsulates control of the radio chip, thereby shielding the application from low-level timing of transmission, interrupt processing, and some details of error detection and message acknowledgment.

One of the first MAC protocols engineered for power control in a WSN is S-MAC [1, 4] (see also T-MAC [2] and Trama [3], which similarly schedule active and sleeping periods). The S-MAC protocol uses active periods of fixed size and sleeping periods of variable size. In S-MAC, nodes are aware of neighbors and exchange schedules and synchronization information with neighbors to adjust sleeping periods. Typical choices in the design of MAC protocols beyond the duty cycle include selecting a technique for resolving contention, and choosing RTS/CTS signaling similar to 802.11. The B-MAC protocol [8] introduces a simpler technique for duty cycling that relies on a hardware trade-off rather than coordinated scheduling. The innovation of B-MAC is Low-Power Listening (LPL), in which nodes sleep for some time, followed by a brief sampling of the radio to detect upcoming transmissions; if there is no radio activity, a node returns to sleep following the sample. LPL requires that transmitters send a preamble to each message such that the length of the preamble is at least the length of the LPL sleeping time. Experiments validate that LPL can achieve a duty cycle of 1% (and if transmit operations are rare, the extra power cost of transmission is tolerable). B-MAC has lower overhead than previous power-efficient MAC protocols, but requires hardware features to implement the preamble (not all radio chips offer this feature). Analysis for a typical sensor network application in [8] reveals that B-MAC's advantage to extend node lifetime is limited: LPL duty cycles cannot be pushed much below 1%. Subsequently, SCP-MAC [5] shows how duty cycles below 0.1% can be obtained, reverting to the theme of scheduled, coordinated sleeping periods rather than using the LPL technique.

Exploiting knowledge at the application layer can enable duty cycles far below 0.1%. An application demonstrated at Intel [10] only requires that nodes report sensor data once per day. Note that arranging all nodes to wake once per day, and also to have a small tolerance interval ϵ, depends on the accuracy of node clocks. It becomes necessary to estimate clock drift [7], which can change significantly over the course of time. At the application layer, duty cycling is useful not only to control radio power, but also to shift sensor coverage where the WSN has sufficiently redundant coverage to allow rotation of sensing duty [6].

Numerous WSN case studies feature scenarios that would enable simple approaches to dealing with temporal partition. For example, a single-hop network (clique topology) eliminates the multi-hop issues. The existence of a gateway or base-station and guaranteed (multi-hop) connectivity to base-station can be leveraged: nodes that lose connectivity to the base-station can revert to a bootstrap protocol, remaining active until connectivity is established. Similarly, the availability of nodes powered by the electric grid can anchor scheduling, since these nodes do not need to sleep. If nodes have access to GPS or some independent source of global time, alignment is easy to obtain. Thus, clock synchronization can be the basis for solving temporal partition, provided nodes are active to exchange synchronization messages.

4 Protocols

We distinguish two approaches for protocol design: (i) the "no cost" approach varies only the length of the sleep period up to the sleep_{upper} bound; (ii) the "probing" approach adds active periods for the purpose of accelerating convergence.

Before we present our protocols in detail, we discuss the issue of how components merge with each other after detection; this pertains to all our protocols. When the active period of a node a overlaps with the active period of a node b, both nodes acquire information about each other. Then using an unspecified "merge rule," either a decides to join b's component or vice versa. There can be a variety of deterministic or randomized merge rules. For example, a node with smaller clock value may join the component of node with larger clock value; alternately, IDs of component leaders may be used to make this decision. Ideally a merge rule should be simple and not impose any communication overhead. Besides the merge rule, there is also the issue of whether or not a node a takes its component along when it joins node b's component. If we assume yes, then node a needs to communicate its decision to the rest of the nodes in its component and this imposes a communication overhead. The advantage of this approach is that for any component, each of its nodes is trying to detect other components, thereby increasing the possibility that this component will join others. In order to minimize communication overhead, our protocols assume that nodes individually make the decision to join other components, without taking the rest of their component along. Exploring the trade-offs between these two approaches is for the future.

4.1 No Cost Approaches

This section presents an efficient "no cost" protocol to solve the temporal partition problem. The basic structure of protocols being considered here is the following.

1. If v receives a message from a node u such that $clock(v) \neq clock(u)$ and if the "merge rule" is satisfied then v copies $clock(u)$ and other associated information.
2. Node v picks its sleep period $s_v \leq \mathsf{sleep}_{upper}$.

The above two steps are executed as part of node v's active period. After completion of its active period, v sleeps for s_v slots and then repeats the above protocol. Due to the assumption that the active period of a node occupies one time slot, the cycle length of node v, $z_v = 1 + s_v$. Variants of the above protocol are obtained by varying how the sleep period s_v is chosen. For example, s_v may vary from node to node and from one cycle to the next, s_v may be chosen using a deterministic rule or a randomized rule, etc. In the rest of this subsection we assume that all nodes in a stably aligned component choose the same sleep period in each cycle, though this may vary from cycle to cycle. This can be achieved without any communication because these nodes share common information (a synchronized clock, the ID of the component leader, etc.) and the sleep period is a function of such information. This is true even if the choice of the sleep period is random; if all nodes in a stably aligned component use their clock value as a seed for the random number generator, they obtain identical sleep periods with no need for additional communication. The following lemma uses elementary number-theoretic arguments to show that the choice of s_v is critical.

Lemma 1. *Let z_a and z_b respectively be the cycle lengths of nodes a and b, in every cycle.*

(i) If z_a and z_b are not relatively prime then there exists an initial displacement of a and b such that the active periods never overlap.
(ii) If z_a and z_b are relatively prime then the active periods of a and b overlap is at most $s_a \cdot s_b$ time slots, regardless of the initial displacement.

Proof. (i) Let $f > 1$ be a common factor of z_a and z_b. For any pair of positive integers m_a and m_b, the quantity $m_a \cdot z_a - m_b \cdot z_b$ is a multiple of f and is therefore distinct from 1. If nodes a and b start with an initial displacement of 1, then no matter how many cycles a runs for and no matter how many cycles b runs for, the active periods of a and b never overlap.
(ii) Without loss of generality, suppose that $z_a > z_b$. Then for any possible

displacement k, the equation $m_a \cdot z_a - m_b \cdot z_b = k$ has a positive integer solutions for m_a and m_b satisfying $m_a \leq s_b$ and $m_b \leq s_a$. Thus the active periods of a and b overlap in at most $s_a \cdot s_b$ time slots.

The above lemma suggests the assignment of relatively prime cycle lengths to different components in order to guarantee convergence. For a pair of components, an easy choice would be integers z and $z+1$ satisfying $2 \leq z < z+1 \leq \mathsf{sleep}_{upper}$ because such a pair is guaranteed to be relatively prime. See Figure 2 for an il-

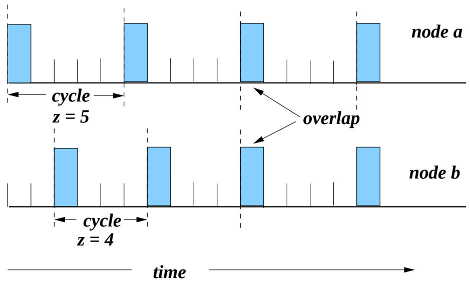


Fig. 2. Illustration for two nodes, a with cycle length 5 and b with cycle length 4. In this example, after the active periods of a and b overlap, node b adopts the cycle length of a and the two nodes will continue to be aligned forever.

lustration of how convergence takes place for $z = 4$. However components are unaware of each other's existence and are therefore unaware of the choices that other components make. Therefore deterministically guaranteeing that the two components make distinct choices is difficult and so we resort to randomization. Let z be a fixed integer satisfying $2 \leq z \leq \mathsf{sleep}_{upper}$. Each node v performs the following step in determining its sleep period.

> $\mathsf{NoCost1}(v)$: In each cycle, node v picks its sleep period s_v uniformly at random from $\{z-1, z\}$.

Thus the cycle length of each node v is either z or $z+1$. Note that the above protocol assumes that all nodes are aware of z. This is justified by assuming that all nodes are informed of sleep_{upper} and use a common, deterministic rule for picking an integer z, $2 \leq z \leq \mathsf{sleep}_{upper} - 1$ (e.g., pick the largest integer z: $2 \leq z \leq \mathsf{sleep}_{upper} - 1$). We now prove an upper bound on the number of time slots it takes for the active periods of a pair of nodes using the NoCost1 rule to overlap. Define the *displacement* of a pair of nodes to be the minimum distance, in time slots, between the active periods of the nodes.

Lemma 2. *In a network in which nodes use the NoCost1 rule for picking sleep periods, for any two nodes a and b, the active periods of a and b will overlap in expected $O(z^3)$ time.*

Proof. Let Z be the random variable denoting the displacement between nodes a and b. Suppose that at the beginning of a cycle $0 < Z < \lfloor z/2 \rfloor$. Then at the end of the cycle, Z increases by 1 with probability 1/4, decreases by 1 with probability 1/4, and retains the same value with probability 1/2. At the two extreme values $Z = 0$ and $Z = \lfloor z/2 \rfloor$, the random variable behaves as follows: (i) if $Z = 0$, then at the end of the cycle, Z increases by 1 with probability 1/2 and retains the same value with probability 1/2; (ii) if $Z = \lfloor z/2 \rfloor$, then at the end of the cycle Z decreases by 1 with probability 1/2 and retains the same value with probability 1/2. This behavior of Z as a function of the number of cycles is a 1-dimensional random walk with reflecting barriers and it is well known [9] that such a random walk will reach $Z = 0$ in expected $O(z^2)$ steps (cycles), which translates to $O(z^3)$ time slots, independent of what the initial value of Z is.

The random walk described in the above proof is a somewhat inefficient way of removing the displacement between a pair of nodes, especially given the fact that if we could somehow force one component to pick z and the other to pick $z+1$ we could get convergence in $O(z)$ cycles ($= O(z^2)$ time slots) because in each cycle the displacement between the two components would *consistently* change by one slot. This observation motivates the following rule for picking a sleep period.

> NoCost2(v): Node v picks s_v uniformly at random from $\{z-1, z\}$ and retains the same sleep period for z cycles.

The use of NoCost2 shaves off a factor of "z" from the expected time to overlap and gets us to within a constant factor of the optimal deterministic selection.

Lemma 3. *In a network in which nodes use the NoCost2 rule for picking sleep periods, for any two nodes a and b, the active periods of a and b will overlap in expected $O(z^2)$ time.*

Proof. Let a *phase* denote z cycles. If a and b pick distinct cycle lengths then they overlap in one phase; the probability of this event is 1/2. On the other hand, with probability 1/2, a and b might pick the same cycle length and continue to have the same displacement at the end of the phase. Therefore, the random variable P, denoting the number of phases before overlap, has the geometric distribution $\text{Prob}[P = k] = 1/2^k$ and therefore $\text{E}[P] = O(1)$. This means that the active periods of a and b overlap in expected $O(1)$ phases, which translates to expected $O(z)$ cycles or expected $O(z^2)$ time slots.

The above lemma considers only two components. We now consider the general case where the network has arbitrarily many stably aligned components of arbitrary sizes. Each node executes NoCost2 and to be concrete we assume the merge rule: if $clock(a) < clock(b)$ then node a joins node b's component.

Theorem 1. *In expected $O(diam(G) \cdot z^2)$ time slots, the network will have exactly one stably aligned component. Here $diam(G)$ refers to the diameter of the network G.*

Proof. Let v be a node with largest clock value. From Lemma 3 we see that in expected $O(z^2)$ time slots, all neighbors of v would have inherited v's clock value. Note that the stably aligned component containing v will continue to have the largest clock value in the network. Continuing inductively, we see that in expected $O(t \cdot z^2)$ time slots, all nodes within t hops from v would have inherited v's clock, leading to the theorem.

Corollary 1. *There is a self-stabilizing protocol that solves the temporal partition problem without using any extra active time periods in $O(diam(G) \cdot z^2)$ time with a $1/z$ duty cycle.*

A cycle length of z and a duty cycle of $1/z$ prevents the nodes from using any extra active time periods, besides the given active periods they are required to have. In such a setting information takes $\Omega(z)$ time slots to traverse $O(1)$ hops in the network and therefore we get an $\Omega(diam(G) \cdot z)$ lower bound on the convergence time for any no cost protocol. The above result is thus a factor of "z" away from this lower bound.

4.2 Probing Approaches

While the no-cost approach shows convergence with no additional power consumption, it is possible to speed up the convergence by increasing the duty cycle by only a constant factor. This is done by permitting nodes to be active for additional time slots outside their active periods for the purposes of "probing" for other components. During the additional activity each node will search for nodes not belonging to its own stably aligned component. Contrary to the no-cost approach, we make the sleep period identical for all nodes in all cycles.

In this section we will consider two methods for probing: deterministic and randomized. In both approaches, during a component's active period, every node will select one probing period from the sleeping period during which it will turn its radio on. This probing period may be selected deterministically or randomly, as we shall see in the following.

Deterministic Probing We starts this section with a deterministic probing protocol. Let z be the cycle length and assume that time slots in a cycle are labeled $0, 1, \ldots, z-1$. For an integer parameter c, $1 \leq c \leq z-1$, we define the *probing period* to be a size-c set of consecutive integers in $[1, z-1]$. The basic structure of the protocol is the following: during its active period, a node chooses its next probing period in a cyclic manner, *i.e.* slots $[1, c]$, then slots $[c+1, 2c]$, and so until the end of its cycle. The node starts again with slots $[1, c]$, and so on. Assuming that each component has an ID which is the greatest ID of any node in the component, we have the following merging rule. During the probing period, if a node v detects a node u which is not in its own component, then if the ID of the u's component is higher than the ID of the v's component, v immediately joins the u's component, otherwise v sends a message containing its clock and its ID's component to u, then u immediately joins the v's component. Recall two assumptions we made in Section 2: (*(i)*) the length of the active period (i.e., 1 time unit) is sufficiently large to allow reliable bidirectional communication and (*(ii)*) the length of overlap between any two time periods is integral. These two assumptions along with the fact that u and v are neighbors implies that all of the communication described above completes before the end of the active period of u. It is straight forward to see that the duty cycle of the protocol is $(1+c)/z$. We now analyze the convergence time of this protocol.

Lemma 4. *The convergence complexity for two nodes is* $O(\frac{z^2}{c})$.

Proof. Observe that each cycles is divided up into $\frac{z-1}{c}$ probing periods, and that following $\frac{z-1}{c}$ cycles every probing period has been selected by both nodes, and thus the two nodes will detect each other. Since the length of one cycles is z time slots the result follows.

Since from Lemma 4 we see that within $O(\frac{z^2}{c})$ time slots every node that is a neighbor of the component with the highest ID will join that component, we can use a similar proof as Theorem 1to get the following result.

Theorem 2. *The convergence complexity of the protocol is* $O(diam \cdot \frac{z^2}{c})$ *where diam is the diameter of the network.*

We now show that the above protocol is optimal to within a constant factor by proving a $\Omega(\frac{z^2}{c} \cdot diam)$ convergence time of any deterministic protocol with duty cycle of $\frac{1+c}{z}$. We say that two connected components *detect each other* if the probing period of one overlaps with the active period of each other. We assume that detection is a sufficient condition to ensure that at least one node belonging to one of the two components will merge with the other.

Define *probing frame* as the union of all probing periods (over many cycles). For instance, if the probing period of a node is $\{1, 2, \ldots, z-1\}$ (the entire sleeping period) each cycle, then the probing frame of the node is the sleeping period itself. In the same way, if the probing period alternates between $\{1, 2\}$ and $\{3, 4\}$, the probing frame is $\{1, 2, 3, 4\}$. Moreover, we assume that the probing frame begins just after the active period. Note that all nodes in one stably aligned component have the same probing frame. The next lemma considers the minimum length of the probing frame in order to guarantee that two connected components will be able to detect each other.

Lemma 5. *Let A and B be two stably aligned components. Regardless of the initial temporal displacement, the active period of A is guaranteed to overlap with the probing period of B or the active period of B is guaranteed to overlap with the probing period of A if and only if the length of the probing frame is at least* $\lfloor \frac{z}{2} \rfloor$ *slots.*

Proof. In this proof we denoted the length of the sleeping period as *sleep* slots and the length of the probing frame as *probe* slots. Recall that one slot is the length of the active period.

Assuming that the active period of A does not overlap with the probing frame of B, this implies that the active period of A starts after the probing frame of B ends. Thus the length of the remaining sleeping period for B is $sleep - probe$, and in order to ensure that the probing frame of A overlaps with the active period of B we require that

$$sleep - probe \leq probe.$$

Note that $z = sleep + 1$, and thus the above equation becomes

$$z - 1 - probe \leq probe \Rightarrow \frac{z}{2} - \frac{1}{2} \leq probe.$$

Since the length of the probing frame must be an integer number of slots, we get that in order for the active period of B to overlap with the probing frame of A we require $probe \geq \lfloor \frac{z}{2} \rfloor$.

The following lemma computes the minimum time complexity for two connected components to detect each other.

Lemma 6. *Given a protocol solving temporal partition using a deterministic probing approach. Let c be the length of each probing period for every node. The minimum time complexity ensured by the protocol for two stably aligned components to detect each other is $\Omega(\frac{z^2}{c})$ slots.*

Proof. Lets assume there are two connected components in the graph, A and B. First note from Lemma 5 that if the length of the probing frame is at least $\lfloor \frac{z}{2} \rfloor$, there exists at least one probing period of A, respectively B, that overlaps with the active period of B, respectively A. Observe that there are $\lfloor \frac{z}{2} \rfloor \cdot \frac{1}{c}$ probing periods in each probing frame, and since one probing period overlaps with an active period, it follows that $\Omega(\frac{z}{c})$ cycles are required before detection is guaranteed. The length of one cycle is z slots, and the result follows.

In the following, we will denote a *component graph* $G_c = (V_c, E_c)$ such that every stably aligned component in the network is a node in V_c and there exists an edge between two nodes A and B in G_c if and only if there exists an edge between at least one node in A and one node in B in the network. We denote $N[A]$ as the set of nodes in the component graph that are within distance 1 of A (which includes A).

Theorem 3. *Consider a protocol that uses the deterministic probing approach to solve the temporal partition problem ensuing a $\frac{1+c}{z}$ duty cycles. Such a protocol has a convergence time $\Omega(\frac{z^2}{c} \cdot diam)$ where diam is the diameter of the network.*

Proof. We are going to make a proof by contradiction. Let us assume that such a protocol exists. In this case, a better complexity should be achievable by this protocol for any component graph. We consider a component graph that is a chain consisting of the components $C_1, C_2, ..., C_{diam}$ where C_i is a neighbor of C_{i-1} and C_{i+1} for every $2 \leq i \leq diam - 1$. We assume that initially every component consists of a single node, and we assign an ID to each node such that $ID(C_1) > ID(C_2) > \ldots > ID(C_{diam})$.

The result of Lemma 6 state that after $\Omega(\frac{z^2}{c})$ slots, a node will detect every neighboring component. Observe that $N(C_i) = \{C_{i-1}, C_i, C_{i+1}\}$ for any $1 \leq i \leq diam$ (within the constraints that $i-1 \geq 1$ and $i+1 \leq diam$). Note that $ID(C_{i-1}) > ID(C_i)$. Thus following $\Omega(\frac{z^2}{c})$ slots C_2 will merge with C_1, C_3 will merge with C_2 etc. Then, after the merging, we obtain a new component graph consisting of the components $C_1 \cup C_2, C_3, \ldots, C_{diam}$ connected in a chain such that $ID(C_1 \cup C_2) > ID(C_3) > ID(C_4) > \ldots > ID(C_{diam})$, since $ID(C_1 \cup C_2) = ID(C_1)$. In this new component graph, the diameter has been reduced by 1. Note that the new component graph retains the same properties as the previous one, and that every component C_3, C_4, ..., C_{diam} still only consists of a single node. This allows us to apply an inductive argument.

We see that one component is removed from the graph each $\Omega(\frac{z^2}{c})$ slots. Thus the convergence time is $\Omega(\frac{z^2}{c} \cdot diam)$, contradicting the initial assumption. Since each node is active during $1+c$ time slots each cycles, we see that the duty cycle is $\frac{1+c}{z}$.

Randomized Probing In this section we will use *randomized probing* to solve the temporal partition. The basic idea of randomized probing is that each node picks one slot at random, from its sleep period, and remains active during that slot. This approach works best when there is a large component and its nodes have picked different slots to probe, thus covering a large fraction of the sleep period. Randomization helps to achieve this and our analysis uses standard arguments similar to those in the "birthday paradox" or the "coupon collector" problem to assert that if the size of a component is a "$\log n$" factor times the length of a cycle, the component will cover the entire cycle with high probability.

To keep exposition simple, we assume that the underlying network is a clique; the technique works for more general constant-diameter, dense graphs. The basic structure of the protocol is the similar to deterministic probing; recall that there we assumed that all nodes have the same cycle length z in all cycles. The key differences are enumerated below.

1. The "merge rule" is different. Node v merges with the component of node u provided u's component is larger in size. If the sizes of the two components are identical, then the IDs of leaders are used to break the tie.
2. Let the time slots in a cycle be labeled $\{0, 1, \ldots, z-1\}$, with 0 denoting the active period. During the active period each node also picks a time slot t uniformly at random from $\{1, 2, \ldots, z-1\}$. After its active period, each node goes to sleep for the next $z-1$ time slots, with the exception of time slot t.

For the purposes of ensuring enough "coverage" of the sleep period, it is critical for this protocol that the random choices of t be independent even for the nodes in the same component.

We start the analysis of this protocol be assuming that there are k components, labeled $C_1, C_2, \ldots, C_k$ such that $|C_1| \geq |C_2| \geq \cdots \geq |C_k|$ and furthermore if $|C_{i+1}| = |C_i|$ then the ID of the component leader of C_{i+1} is greater than the ID of the leader of C_i. Note that this labeling is just for the purposes of the proof is not computed by the algorithm. This ordering of components guarantees that if a node $v \in C_i$ leaves its component to join another, then that component is one of $C_1, C_2, \ldots, C_{i-1}$. Our analysis assumes that $z \leq n/8 \log n$ (used in the proof of Lemma 7). There are two cases depending on the size of C_1.

"Large" C_1. Suppose that $|C_1| \geq 2z \cdot \log n$. We show that in this case, with high probability, in one cycle, all nodes in $C_2 \cup C_3 \cup \cdots \cup C_k$ will join component C_1. Consider a node $v \in C_2 \cup C_3 \cup \cdots \cup C_k$. The probability that v's extra active period will not overlap with the active periods of any node in C_1 is

$$\left(1 - \frac{1}{z}\right)^{|C_1|} \leq \left(1 - \frac{1}{z}\right)^{2z \cdot \log n} \sim e^{-2 \log n} = \frac{1}{n^2}.$$

Since there are n nodes, by using the union bound we see that with probability at most $1/n$ there is a node $v \in C_2 \cup C_3 \cup \cdots \cup C_k$ whose extra active period does not overlap with the extra active period of any node in C_1. Therefore, with probability at least $1 - 1/n$ every node outside C_1 will join C_1 in one cycle (of length z time slots).

"Small" C_1. Here we suppose that $|C_1| < 2z \cdot \log n$. In this case we show that the number of components decreases by a constant fraction in each cycle. Consider a permutation π of all nodes in which C_1 comes first, followed by C_2, followed by C_3, and so on. The nodes in each C_i appear in some arbitrary order in π. Let v be the node with rank $\lfloor n/2 \rfloor$ in π and let C_j be the component that contains v. Call C_j the *middle component*, $C_1, C_2, \ldots, C_{j-1}$, the *big components*, and $C_{j+1}, C_{j+2}, \ldots, C_k$, the *small components*. We will show two properties.

Property 1. The number of small components is at least a constant fraction of the number of large components.

Property 2. In one cycle, all the small components will disappear, with high probability.

Together these properties lead to the claim that in the "small" C_1 case a constant fraction of the components disappear in each cycle. Property (2) follows from the same argument that was used to deal with the "large" C_1 case. Property (1) is proved in the following lemma.

Lemma 7. *The number of small components is at least a constant fraction of the number of large components.*

Proof. Since $|C_1| \leq 2z \log n$, every component has size at most $2z \log n$. Using the assumption that $z \leq n/8 \log n$ yields an upper bound of $n/4$ on the size of every component. From this it follows that the union of the small components has size at least $n/4$. Let the number of small components be s. The average size of the small components is at least $n/4s$. Therefore every big component has size at least $n/4s$, implying that the number of big components is at most $2s$.

Note that if the number of components is $n/2z \log n$ or fewer, then there is at least one component of size at least $2z \log n$. Even if the network starts off with n components (i.e., every node is a components by itself), the progress we make in the "small" C_1 case implies that in $O(\log(z \log n))$ cycles, we will reach the "large" C_1 case. Given that convergence takes one additional cycle (with high probability) and given that each cycle has z time slots, we get the following theorem.

Theorem 4. *In expected $O((\log z + \log \log n) \cdot z)$ time slots, the network will have exactly one stably aligned component.*

To compare this convergence time with the convergence time of the no cost approach proved in Theorem 1, first note that since the network is assumed to be a clique, the no cost approach guarantees $O(z^2)$ convergence time. It is easy to compare $O((\log z + \log \log n) \cdot z)$ with $O(z^2)$ and note that whenever $\log \log n = o(z)$, we get an asymptotically faster convergence time using randomized probing. Informally speaking, unless z is very small, the randomized probing approach is much faster than the no cost approach for dense network.

5 Conclusions

The "no cost" approach provides an $O(diam(G) \cdot z^2)$ convergence time with $1/z$ duty cycle for any $z \leq \mathsf{sleep}_{upper}$. The optimality of this result is currently unclear to us and it is possible that further randomization could improve the convergence time to $O(diam(G) \cdot z \log z)$. The deterministic probing approach yields a spectrum of convergence times and duty cycles, obtained by varying c relative to z such that the product of the convergence time and duty cycle equals $O(diam(G) \cdot z)$. For example, if c is picked close to $z^{1/2}$, we get a convergence time of $O(z^{3/2})$ and a duty cycle that is approximately $1/z^{1/2}$. While this flexibility might seem like an advantage that the deterministic probing approach has over the "no cost" approach, it is worth pointing out that we can pick any $z \leq \mathsf{sleep}_{upper}$ for the "no cost" approach and obtain a similar flexibility. In the light of this, it is not clear if the deterministic probing approach has any advantage over the "no cost" approach. However, the randomized probing approach does yield asymptotically faster convergence (with the same duty cycle) relative to the "no cost" approach under certain circumstances. If the network is assumed to be a clique, then the "no cost" approach guarantees an $O(z^2)$ convergence time for a $1/z$ duty cycle. Comparing this with the $O((\log z + \log \log n) \cdot z)$ convergence time of the randomized probing approach, we note that whenever $\log \log n = o(z)$, we get an asymptotically faster convergence time using randomized probing. Informally speaking, unless z is very small, the randomized probing approach is much faster than the no cost approach for dense network. Given this positive news for the randomized probing approach, it may be worthwhile to expand this approach to more general classes of graphs. For example, we are currently analyzing the randomized probing approach for classes of graphs whose min-cut value is bounded from below.

References

1. W Ye, J Heidemann, D Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOMM'02: Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communication Societies*, 2002.
2. T van Dam, K Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *SenSys'03: Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, pp. 171-180, 2003.
3. V Rajendran, K Obraczka, JJ Garcia Luna Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys'03: Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, pp. 181-193, 2003.
4. W Ye, J Heidemann, D Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE Transactions on Networking*, 2004.
5. W Ye, F Silva, J Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *SenSys'06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pp. 321-334, 2006.
6. Q Cao, T Abdelzaher, T He, J Stankovic. Towards optimal sleep scheduling in sensor networks for rare-event detection. In *IPSN'05: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.
7. S Ganeriwal, D Ganesan, H Sim, V Tsiatsis, MB Srivastava. Estimating clock uncertainty for efficient duty-cycling in sensor networks. In *SenSys'05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pp. 130-141, 2005.
8. J Polastre, J Hill, D Culler. Versatile low power media access for wireless sensor networks. In *SenSys'04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 95-107, 2004.
9. R Motwani and P Raghavan. *Randomized Algorithms.* Combridge University Press, New York(NY), 1995.
10. N Ramanathan, M Yarvis, J Chhabra, N Kushainagar, L Krishnamurthy, D Estrin. A stream-oriented power management protocol for low duty cycle sensor network applications. In *EMNETS'05: Proceedings of the Second IEEE Workshop on Embedded Networked Sensors*, 2005.
11. Y Li, W Ye, J Heidemann. Energy efficient network reconfiguration for mostly-off sensor networks. In *SECON'06: Proceedings of the Third IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 527-535, 2006.