Probabilistic Methods For Dynamic Node Population Management In Sensornets

Jonathan Tate and Iain Bate

November 2009

Abstract

Sensornets provide coverage of physical phenomena over extended periods, perhaps months or years. However, active nodes may deplete finite batteries within days, and are prone to failure. The sensornet application may require a given number of active nodes within each region to provide appropriate sensor redundancy and processing capacity. If many nodes are deployed, at any given time a smaller working set of the correct size can be selected for duty. In this paper we present a lightweight approach to active population management. An omniscient overview of network state is not required, and expensive communication activity is minimised. Probabilistic methods are employed, ensuring that individual nodes can make appropriate decisions using only locally available information.

1 Introduction

Sensornets compose many small, low-cost computing nodes into distributed systems deployed into physical environments of interest. Nodes have restricted energy, computation and storage resources and therefore limited utility in isolation; co-operation and co-ordination is necessary to address realistic problems. Sensornet applications may require a minimal number of active nodes to provide the sensor redundancy and processing capacity required for optimal performance.

Suppose the application functions optimally with cells of n nodes, but the physical distribution of nodes is such that a given geographic cell region contains $m > n$ nodes. Too many or too few *active* nodes may be detrimental to application behaviour. Nodes not currently required to participate should be reserved for future selection and enter low-power *inactive* states.

Some mechanism is required to regulate which nodes actively participate in the network. Managing large node populations distributed over large physical regions is difficult. Tight resource constraints imply the overheads of stateful protocols may be unaffordable.

Consider some given cell containing m nodes. A static assignment of n nodes as *active* and $m - n$ nodes as *inactive* achieves the correct number of *active* nodes but is inflexible. If one or more *active* nodes should fail then this correctness property no longer holds. This is likely in sensornets deployed in hazardous environments, or composed from unreliable nodes. If the same subset of nodes is active at all times, and cycled to exhaustion despite the presence of the additional $m - n$ *inactive* nodes, then network lifetime is bounded by the lifetime of individual nodes.

If the $m - n$ surplus can be considered as a pool of *spares*, a suitable mechanism can dynamically assign n nodes as *active* at any given time, with a different set of n nodes being drawn from the pool of m nodes as time progresses. This allows the fairness property that the duty burden is shared evenly between all m nodes, while retaining the correctness property that n nodes actively participate in the network at all times. Excessive churn is prevented explicitly by damping state transition rates and randomising candidate selection.

If a cell contains exactly $m = n$ nodes then the control mechanism should maintain n nodes active at all times. If a cell contains $m < n$ nodes then it is not possible for any control mechanism to achieve a population of n simultaneously active nodes, but keeping all m nodes active at all times yields behaviour as close as possible to the desired behaviour.

The lifetime of a single node no longer restricts the network lifetime, as a failed node is replaced in time by another node drawn from the set of spares. The operator can add more nodes after the sensornet begins operation to extend the network lifetime, enabling periodically replenished sensornets to operate perpetually.

This paper is organised as follows. Section 2 discusses related work. Section 3 defines basic protocol aims and concepts. Section 4 defines the state machine, and section 5 defines state transitions. Sections 6 and 7 discuss algorithm overheads. Section 8 presents experimental results. Section 9 presents conclusions.

2 Related work

In a typical sensornet it is rare for all nodes to perform useful work at all times. Careful node state management, placing network subsets in low-energy inactive states when not required to actively participate, significantly improves energy efficiency [3]. Communication, computation and storage overheads of optimal algorithms may, unfortunately, exceed the savings [2].

The *Random Asynchronous Wakeup* protocol [6] implements a randomised and distributed algorithm. Nodes make local decisions on whether to sleep or remain awake. Within each time frame each node is awake for a randomly chosen fixed interval. An integrated routing protocol selects next-hop nodes

from a set of equivalent next-hop locations, with only probabilistic guarantees that at least one is awake.

Similar functionality is provided by the *Asynchronous Random Sleeping* scheme [4] which is principally useful where no inter-node coordination is possible. However, such scenarios might be considered unusual as sensornets generally execute distributed and cooperative sensing and processing applications.

The *Probing Environment and Adaptive Sleeping* protocol [12] implements an adaptive sleep policy in which nodes sleep for an exponentially distributed duration then wake and transmit a probe message. If any nearby nodes happen to be awake they transmit a reply message. If any such reply message is received the node is not required at this time and sleeps again; otherwise, it remains awake until it fails or runs out of energy. A significant weakness is that when a node fails there is zero local network coverage until some other nearby node wakes with indeterminate delay. If the failed node has accumulated significant data this cannot be replaced by that of other nearby nodes.

The *Lightweight Deployment-Aware Scheduling* algorithm [11] aims to improve network energy efficiency by switching off redundant nodes without access to accurate location or directional information. Observing that nodes require up to 11 active neighbours to provide a 90 percent chance of complete redundancy, LDAS allows network designers to trade-off sensing redundancy against energy consumption. This protocol is most appropriate and efficient in networks where most nodes are required for physical sensing rather than for distributed data processing.

The *Cyclic Duty Allocation Protocol* [9] manages a sensornet cell such that a fixed number of nodes are active at any given time, equally sharing costs and responsibilities between all nodes. However, it assumes nodes never fail or move between cells, assumes cells always contain the correct number of nodes, and does not allow unused spare nodes to sleep for long periods.

Application-aware traffic scheduling [5] is an alternative approach to maximising energy efficiency and hence network lifetime. *Multi-Sensornets* applies a genetic algorithm to balance nodes' energy consumption in a distributed data fusion application. Peer nodes coordinate dataflow schedules such that the time during which they are required to be awake is minimised, allowing nodes to safely sleep at other times without disrupting network coverage.

3 Active Duty Control Protocol

The *Active Duty Control Protocol* (ADCP) regulates the population of a network cell to maintain an active population of fixed size. If too many

nodes are active, some are made inactive. If too few nodes are active, some of the pool of spares are made active. We assume nodes form a connected graph within each cell.

ADCP is somewhat similar in principle to PEAS [12], but more sophisticated. ADCP defines network regions by fixed geographical cells rather than inconsistent wireless connectivity. ADCP maintains active populations, which can be > 1, within each region, whereas PEAS activates only a single active node within each region. Unlike PEAS, a failed node is quickly replaced, and remaining nodes can provide reduced but non-zero capability prior to replacement. Non-active nodes never transmit, saving energy and preventing transmission clashes. Fair wear balancing is implemented.

Each node knows the target active cell population n, and the total number of nodes available for selection for active duty m, but does not know the current active cell population δ. For a cell of total population $l \geq m$, there exist $m - l$ nodes not currently available for selection, which may later replace failed nodes.

Each node acts independently, and each decision is made without reference or consultation with other nodes. Information about other nodes within a network or network cell is not actively *shared* or *solicited*, but instead is *learned* by observation. Non-active nodes do not transmit in the wireless medium. This reduces overhead and wireless communications clashes.

3.1 Periodic signal generation

ADCP requires that all nodes transmit at least once within each system *epoch* of length e. These transmissions can be anonymous, and may occur implicitly as part of normal application behaviour. In this paper we assume a periodic signal generation mechanism called *Lightweight Improved Synchronisation Primitive* (LISP); we summarise the definition here, but the full definition and analyses are given in [10].

Assume a network cell consists of a set Σ of nodes $S_1 \cdots S_n$ where $n \geq 2$. If $n = 1$, there is obviously no need for inter-node coordination; the algorithm will function correctly but will do nothing. Each node S_i acts independently but shares an identical set of behavioural rules. The running time of the system is divided into a set of system *epochs* of equal period e such that $\forall j : E_j = e$. The sequence of system epochs E_j is defined by the natural ordering of $j \in \mathbb{N}$.

Within each system epoch E_j it is required that each node $S_i \in \Sigma$ shall execute a single instance of a periodic synchronisation event V_i exactly once. These events are used only by the protocols described in this paper, and are not related to any events used by the sensornet application. All events V_i are periodic with identical period $p_i = e$. The occurrence of a specific event at a specific node i within a specific system epoch j is labelled V_{ij}. It is required that all events V_{ij} are executed within epoch E_j.

Each node $S_i \in \Sigma$ has a local clock used to measure the *local phase* ϕ_i which increases from 0 to ϕ_{max} in time $p_i = e$. When $\phi_i = \phi_{max}$ at node S_i in epoch E_j, node S_i transmits a *synchronisation message* and resets its local phase as $\phi_i = 0$, corresponding to event V_{ij}. No global clock is required. Peer nodes $T \in (\Sigma \setminus S_i)$ receive the V_{ij} synchronisation message at their local phase ψ_{ij} but do not know the identity of S_i.

Within every epoch E_j all nodes S_i record the local phase of peer node synchronisation messages, using this information to modulate their local phase to coordinate behaviour within the cell. *Desynchronisation* protocols maximise the time between synchronisation events for all nodes in a given epoch, converging on an *equilibrium state* in which synchronisation events occur spaced evenly in time [7].

Each node S_i records the local phase of the peer synchronisation events $V_{i\beta}$ and $V_{i\gamma}$, occurring immediately before and immediately after the local synchronisation event V_i respectively, and discarding all others. The corresponding peer nodes $S_{i\beta}$ and $S_{i\gamma}$ are labelled the *phase neighbours* of node S_i. The phase difference between V_i and $V_{i\beta}$ is calculated as $\phi_{i\beta}$, and the phase difference between V_i and $V_{i\gamma}$ is calculated as $\phi_{i\gamma}$. Note that $\phi_{i\beta}$ is always negative and $\phi_{i\gamma}$ always positive owing to natural ordering of events.

The *phase error* $\theta_i = \phi_{i\beta} + \phi_{i\gamma}$ is the phase difference between the local synchronisation event V_i and the target midpoint of phase neighbour synchronisation events $V_{i\beta}$ and $V_{i\gamma}$. Each node S_i alters its local phase by $\Delta\phi_i = -f_\alpha \theta_i$ upon observing $V_{i\gamma}$, where $f_\alpha \in (0, 1]$ is the *feedback proportion* governing the balance between responsiveness and stability. Given an otherwise unchanging network, $\forall i : ||\Delta\phi_{ij}|| \to 0$ as $j \to \infty$ in successive epochs. Observe that $\forall i : \theta_i = 0$ in the *desynchronised equilibrium state*. If phase error $\theta_i = 0$ then phase change $\Delta\phi_i = 0$ also; no action is required when the system has converged.

Algorithm 1 defines behaviour at all nodes $S_i \in \Sigma$.

3.2 Cell subpopulations

Each node of the network population is a member of exactly one subpopulation. The *active set*, Δ, contains nodes which are currently active within the network and are available for participation in wireless communication. The *reserve set*, Λ, contains nodes which are not currently active within the network, but which may become active in the future if required. The *inactive set*, Γ, contains nodes not currently participating.

Nodes in Γ neither transmit nor receive messages in the wireless medium. Nodes in Λ may passively listen to the wireless medium from time to time, but are not assumed to be listening at all times; when not listening, radio modules can be switched into low-power modes. Nodes in Λ do not transmit, except to announce leaving Λ to join Δ as described in section 4.

Algorithm 1 : LISP executing at node S_i

Require: Observed predecessor sync phase, $\phi_{i\beta} = nil$
Require: Observed successor sync phase, $\phi_{i\gamma} = nil$
while monitoring local phase ϕ_i increasing over time **do**
  if sync event $\neq V_i$ observed **then**
    if $\phi_{i\gamma} = nil$ **then**
      $\phi_{i\gamma} \Leftarrow \phi_i$
      if $\phi_{i\beta} \neq nil$ **then**
        $\theta_i \Leftarrow \phi_{i\beta} + \phi_{i\gamma}$
        $\Delta\phi_i \Leftarrow -f_\alpha \theta_i$
        $\phi_i \Leftarrow (\phi_i + \Delta\phi_i) \bmod \phi_{max}$
        $\phi_{i\gamma} \Leftarrow (\phi_{i\gamma} + \Delta\phi_i) \bmod \phi_{max}$
      end if
    else
      $\phi_{i\gamma} \Leftarrow \phi_i$
    end if
  end if
  if $\phi_i \geq \phi_{max}$ **then**
    if $\phi_{i\beta} = nil$ **then**
      $\phi_{i\beta} \Leftarrow \phi_{i\gamma}$
    end if
    $\phi_{i\gamma} \Leftarrow nil$
    $\phi_i \Leftarrow 0$
    fire own sync event V_i
  end if
end while

At any given time the activity distribution of the population is such that γ nodes are in the *inactive set*, Γ, λ nodes are in the *reserve set*, Λ, and δ nodes are in the *active set*, Δ. The total number of nodes which are available, but not necessarily selected, for active duty is given by $m = \lambda + \delta$ at any given time. We need only δ, m and n to calculate *state change probabilities*.

During any arbitrary period of length e each active node broadcasts exactly one synchronisation packet. Each node counts synchronisation packets, d, observed within a LISP epoch. d estimates the number of active peers. $\delta = d + 1$ estimates the total active subpopulation size, including the node making the estimate. If the value of m is known then it is trivial to find $\lambda = m - \delta$, such that the proportion of *active* nodes is $\frac{\delta}{m}$ and the proportion of *non-active* nodes is $\frac{\gamma+\lambda}{m}$.

4 ADCP protocol states

Figure 1 illustrates the ADCP states in UML statechart format. There are three composite states in which a given node can exist, corresponding to

the three cell subpopulations described in section 3.2. Nodes in the *inactive* composite state are members of the *inactive set*, Γ. Nodes in the *reserve* composite state are members of the *reserve set*, Λ. Nodes in the *active* composite state are members of the *active set*, Δ.

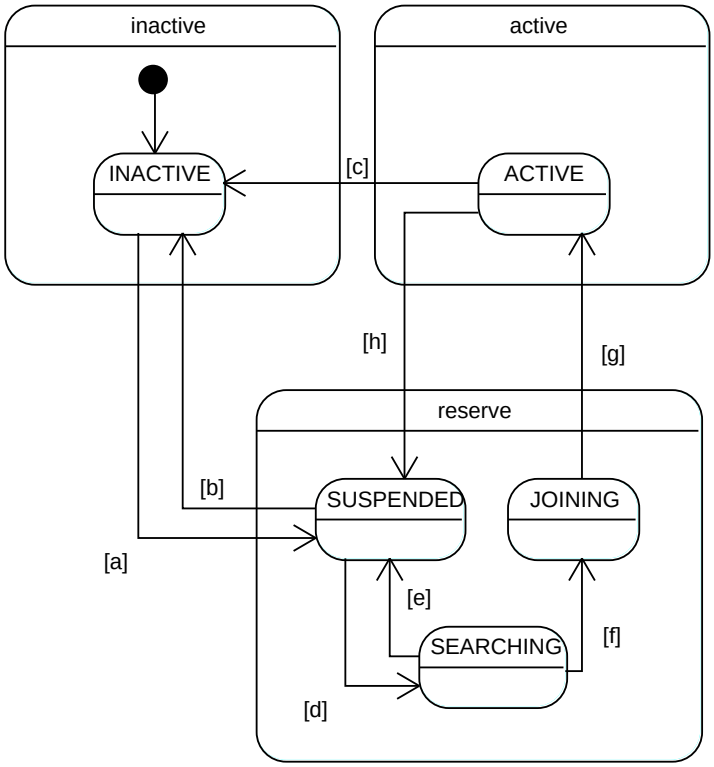

Figure 1: Finite State Machine for ADCP states

4.1 Substates of the *inactive* composite state

INACTIVE - Node does not participate in network activity confined to the *ACTIVE* state, and is not a candidate for selection to make up the shortfall in an underpopulated network cell.

Each node starts in the *INACTIVE* state. The only possible transition is *INACTIVE→SUSPENDED* which moves the node into the *reserve set*, Λ, making the node a candidate for becoming an active participant.

Nodes can stay in the *INACTIVE* state for any arbitrary duration as determined appropriate by the sensornet designer. If the node *mean time to failure* (MTTF) is known, the release of nodes via the *INACTIVE→SUSPENDED* transition can be controlled such that candidates enter the *reserve set*, Λ, at a similar rate to failed nodes leaving the sensornet.

4.2 Substates of the *reserve* composite state

SUSPENDED - A node in the *reserve set*, Λ, is not currently active but is reserved for future selection.

SEARCHING - A node listens to the wireless medium for a period of equal length to one system epoch, to determine how many peers are in the *active set*, Δ. After this period, the node must transition to *JOINING* if able to rejoin Δ, or otherwise return to *SUSPENDED* and remain in the *reserve set*, Λ.

JOINING - A node waits for an opportunity to rejoin the *active set*, Δ. The transition occurs as soon as a suitable opportunity arises, or after waiting for the duration of one epoch if no suitable opportunities arise, for example in an otherwise empty cell.

The main difference between *SUSPENDED* and *INACTIVE* is that nodes in the former are eligible to become members of the *active set*, Δ, to push a cell toward the desired state where $\delta = n$. Each *SUSPENDED* node measures the progress of time. After each duration of length e, the node randomly decides with *suspended-searching probability*, p_ω, to undergo the *SUSPENDED*→*SEARCHING* transition; otherwise, it simply remains in the *SUSPENDED* state.

A node entering the *SEARCHING* state remains as such for duration e. During this time, the node estimates δ, the cardinality of the *active set*, Δ, and determines whether its is *underpopulated*. If false, the *SEARCHING*→*SUSPENDED* transition is always taken and the node remains in the *reserve set*, Λ. If true, the node *may* join the *active set*, Δ; the probability p_ψ of taking this action depends on the extent of *underpopulation*. If this probabilistic test succeeds the node takes the *SEARCHING*→*JOINING* transition, and will eventually join the *active set*, Δ. Otherwise it takes the *SEARCHING*→*SUSPENDED* transition and remains in the *reserve set*.

A node entering the *JOINING* state listens for *synchronisation pulses* from other nodes. Each node in Δ broadcasts a pulse in each period of length e. When the node observes such a pulse it immediately undertakes the *JOINING*→*ACTIVE* transition, joining the *active set* and signalling this to its new *ACTIVE* peers by transmitting its own LISP pulse. If no such pulses be observed after waiting for duration e we conclude $\delta = 0$; the node immediately undertakes the *JOINING*→*ACTIVE* transition.

4.3 Substates of the *active* composite state

ACTIVE - Node is in the *active set* Δ. At the end of each epoch the node either stays in Δ, or elects to transition to *SUSPENDED*, joining the *reserve set* Λ.

Nodes in the *ACTIVE* state are active participants in the network, implementing whatever network and computation duties are required by the distributed application running within the sensornet. This is the only ADCP state which is not an energy-saving state.

If the characteristics of the sensornet prioritise an equitable distribution of responsibility over active population stability, we can arrange for *ACTIVE* nodes to occasionally elect to undertake the *ACTIVE→SUSPENDED* transition from the *active set*, Δ, to the *reserve set*, Λ. Each *ACTIVE* node independently decides, with period e, whether to undertake the *ACTIVE→SUSPENDED* transition with *voluntary suspension probability*, p_τ, or to remain *ACTIVE*.

Assuming a node elects to undertake the *ACTIVE→SUSPENDED* transition, a member of the Λ will then take its place by the normal mechanism to correct an *underpopulated* cell. The node which has just joined Λ may rejoin Δ at some point; perhaps immediately, unless specifically prohibited.

5 Probabilistic state transitions

Each node has an equivalent view of the cell and will independently calculate the same probability p for each nondeterministic state transition. The number of nodes which independently decide to undergo a given state transition, x, observes a *binomial distribution* [1]. x is distributed as $x \sim B(y, p)$ where y nodes are eligible to undergo a given transition and p is the probability that an eligible node independently elects to do so.

We wish to obtain a situation in which the cell population δ equals the target population n. n is the target *active* population, δ is the current *active* population, and m is the total usable *active* and *reserve* population. $\epsilon = \delta - n$ is the desired *active* population change. Cells can be *overpopulated*, *underpopulated*, or *correctly populated*. Nodes estimate cell populations, and calculate p for each feasible nondeterministic transition.

If each node uses an independent random number source, seeded using entropy sources such as physical sensors, and x observes a binomial distribution, the *expected value* is $E(x) = yp$ with variance $V(x) = yp(1-p)$ [1]. We calculate p such that $yp = |\epsilon|$. The *expected value* of x within any epoch is the number of nodes which must undergo state transition to restore the *active set* cardinality to the desired value $|\Delta| = n$. By the *Law of Large Numbers* [1], the average value of x tends to approach, and stay close to, the binomial distribution *expected value* across multiple epochs.

5.1 Overpopulated cell: $\delta > n$

The *ACTIVE→SUSPENDED* state transition is required to correct *overpopulation*. All members of Δ are eligible in any given system epoch. It follows that $x \sim B(y_a, p_\chi)$ where the number of eligible *ACTIVE* nodes

$y_a = \delta$. p_χ is the probability that a node independently elects to take the *ACTIVE→SUSPENDED* transition. We require p_χ such that $y_a p_\chi = |\epsilon|$. A trivial rearrangement yields $p_\chi = \frac{|\epsilon|}{\delta}$ where $y_a = \delta$. $p_\chi = 0$ unless the cell is *overpopulated*.

$\pi_\chi \in (0, 1]$ is the *node suspension coefficient*, which is a scaling factor applied to p_χ to reduce the rate at which nodes move from the *active set*, Δ, to the *reserve set*, Λ. To utilise the *node suspension coefficient* we extend the definition of p_χ given above to $p_\chi = \pi_\chi \frac{|\epsilon|}{\delta}$. As $\pi_\chi \neq 0$ it is always possible for the cell to progress toward the target active population n, and as $\pi_\chi \leq 1$ the *expected value* $E(x) = \pi_\chi \frac{|\epsilon|}{\delta} y_a = \pi_\chi |\epsilon|$ never exceeds that of the simple case in which $\pi_\chi = 1$.

Given $\delta > n$, it is trivially always true that there are sufficient candidates in Δ which can undertake the *ACTIVE→SUSPENDED* transition, so it is always true that we can find a suitable $p_\chi \in (0, 1]$ such that the *expected value* $E(x) = |\epsilon|$.

5.2 Underpopulated cell: $\delta < n$

The *SEARCHING→JOINING→ACTIVE* state transition sequence is required to correct *underpopulation*. The proportion of *SEARCHING* nodes in Λ is p_ω (see section 4.2). All *SEARCHING* members of Λ are eligible in any given system epoch. It follows that $x \sim B(y_s, p_\psi)$ where $y_s = \lambda p_\omega$ and $\lambda = m - \delta$.

p_ψ is the probability that a node independently elects to take the first part of the sequence, the *SEARCHING→JOINING* transition, which inevitably leads to the second part, the *JOINING→ACTIVE* transition. We require p_ψ such that $y_s p_\psi = |\epsilon|$. A trivial rearrangement yields $p_\psi = \frac{|\epsilon|}{(m-\delta)p_\omega}$ where $y_s = \lambda p_\omega$. $p_\psi = 0$ unless the cell is *underpopulated*.

$\pi_\psi \in (0, 1]$ is the *node activation coefficient*, which is a scaling factor applied to p_ψ to reduce the rate at which nodes move from the *reserve set*, Λ, to the *active set*, Δ. We extend the definition of p_ψ given above to $p_\psi = \pi_\psi \frac{|\epsilon|}{(m-\delta)p_\omega}$. As $\pi_\psi \neq 0$ it is always possible to progress toward the target $\delta = n$, and as $\pi_\psi \leq 1$ the *expected value* $E(x) = \pi_\psi \frac{|\epsilon|}{(m-\delta)p_\omega} y_s = \pi_\psi |\epsilon|$ never exceeds that of the simple case in which $\pi_\psi = 1$.

Given $\delta < n$, there may be insufficient candidates for $E(x) = |\epsilon|$ as $y_s < |\epsilon|$ implies $p_\psi > 1$. As probability $p_\psi \in (0, 1]$ we define $p_\psi = \max(\pi_\psi \frac{|\epsilon|}{(m-\delta)p_\omega}, 1)$. If $\lambda p_\omega < |\epsilon|$, forcing $p_\psi = 1$ in a uniform distribution, the *expected value* is trivially $E(x) = \lambda p_\omega$. The *expected value* of the *active set* population is $\delta < n$ after a single epoch, although $\delta \to n$ in successive epochs as time $t \to \infty$ provided that $(\lambda + \delta) \geq n$.

5.3 Correctly populated cell: $\delta = n$

All *ACTIVE* nodes are eligible to undertake the *ACTIVE→SUSPENDED* transition in any given epoch. $p_\tau \in [0, 1]$ is the probability that an *ACTIVE* node independently elects to undertake the transition, and is specified rather than calculated. It follows that $x \sim B(y_a, p_\tau)$ under a binomial distribution where the number of eligible *ACTIVE* nodes $y_a = \delta$. $p_\tau = 0$ unless the cell is *correctly populated*.

Active population volatility correlates with p_τ. Lower values favour stable populations, and higher values favour equitable burden distribution. If we wish the average number of nodes moving from Δ to Λ per system epoch to be z, we set the required *expected value* of the distribution as $E(x) = z$. As $E(x) = y_a p_\tau$, and $y_a = \delta$, we can extract $p_\tau = \frac{z}{\delta}$.

Setting $z < (m - n)p_\omega$ prevents the population of Δ deviating far from the ideal $\delta = n$ for long durations, while allowing sufficient interchange between Δ and Λ for equal burden distribution in the long term.

5.4 State transition guard conditions

Table 1 defines guard conditions for the ADCP state transitions shown in figure 1. Transitions with guard definitions containing a random number, r, are non-deterministic; all other transitions are deterministic.

Transitions $\{a, b, c\}$ relate to nodes joining or leaving a sensornet cell. Mobile nodes may *become available* or *cease to be available* when crossing cell boundaries, or when replenishing depleted energy supplies.

Transitions $\{d, e, f, g, h\}$ relate to nodes currently eligible to be selected as *ACTIVE*. At all such nodes, subpopulation estimation methods defined in section 3.2 find the population error estimate ϵ, and probability parameters p_ω, p_ψ, p_χ and p_τ defined in section 5 are calculated. A random number $r \in [0, 1]$ is generated when a probabilistic decision is required, and compared against the calculated probability p; if $r \leq p$, the transition is possible and is undertaken.

Label	Condition
a	*node becomes available*
b	*node ceases to be available*
c	*node ceases to be available*
d	$r > p_\omega$
e	$\epsilon \geq 0$
f	$\epsilon < 0 \wedge r > p_\psi$
g	$\epsilon < 0$
h	$(\epsilon = 0 \wedge r > p_\tau) \vee (\epsilon > 0 \wedge r > p_\chi)$

Table 1: Guard conditions for ADCP FSM

6 Cost analysis

Algorithm 2 defines the behaviour of ADCP executing at each node $S_i \in \Sigma$, using the probabilistic state transitions defined in section 5 and cell subpopulation estimation methods defined in section 3.2. ADCP is a lightweight protocol. The only required data storage is for a single integer at each node, used to store the estimate of *active set* cardinality, δ, upon which probabilistic state transition decisions are based. It follows that ADCP storage cost is O(1) in node population.

For a *SEARCHING* node, the δ estimate is incremented every time another cell member broadcasts a *sync pulse*. Under ADCP only the *ACTIVE* and *JOINING* nodes broadcast these, once per epoch, so algorithmic cost is O(n) in cell population. Other ADCP behaviour, such as probabilistic state transitions, occur at each node exactly once per epoch and are independent of population size; ADCP is O(1) in size for these, but O(1) is subsumed by O(n).

Algorithm 2 : ADCP executing at node S_i

Require: Target cell population, n
Require: Suspended-Searching probability, p_ω
Require: Active-Suspended probability, p_τ
Require: Current ADCP state s_i =*SUSPENDED*

while monitoring local phase ϕ_i increasing over time **do**
  if $\phi_i = \phi_{max}$ **then**
    if s_i =*SUSPENDED* **then**
      generate random number $r \in [0, 1]$
      if $r > p_\omega$ **then**
        $s_i \Leftarrow$ *SEARCHING*
      end if
    else if s_i =*SEARCHING* **then**
      estimate current / target active pop. error, ϵ
      if $\epsilon \geq 0$ **then**
        $s_i \Leftarrow$ *SUSPENDED*
      else
        calculate Searching-Joining probability, p_ψ
        generate random number $r \in [0, 1]$
        if $r > p_\psi$ **then**
          $s_i \Leftarrow$ *JOINING*
        end if
      end if
    else if s_i =*JOINING* **then**
      fire own synchronisation event at node S_i
      reset $\phi_i = 0$
      $s_i \Leftarrow$ *ACTIVE*
    else if s_i =*ACTIVE* **then**
      estimate current / target active pop. error, ϵ
      if $\epsilon > 0$ **then**
        calculate Active-Suspended probability, p_χ
        generate random number $r \in [0, 1]$
        if $r > p_\chi$ **then**
          $s_i \Leftarrow$ *SUSPENDED*
        end if
      else if $\epsilon = 0$ **then**
        generate random number $r \in [0, 1]$
        if $r > p_\tau$ **then**
          $s_i \Leftarrow$ *SUSPENDED*
        end if
      end if
    end if
  else if s_i =*JOINING* **then**
    if other node $S_j \neq S_i$ has just broadcast **then**
      fire own synchronisation event at node S_i
      reset $\phi_i = 0$
      $s_i \Leftarrow$ *ACTIVE*
    end if
  end if
end while

7 Energy consumption

For a cell containing l nodes in total, we have cell subpopulations Γ, Λ and Δ such that $\gamma + \lambda + \delta = l$. Within each subpopulation we assume nodes have a similar average rate of energy consumption. We label these energy consumption rates w_Γ, w_Λ and w_Δ for subpopulations Γ, Λ and Δ respectively.

We can estimate the energy consumption rate of a cell at a given time t using subpopulation sizes and the average rate of energy consumption of a node within a subpopulation. The total rate of energy consumption across a stable network cell, w_{cell}, is given in equation 1. To obtain total energy consumption, multiply w_{cell} by the number of elapsed time units.

$$w_{cell} = \gamma_t w_\gamma + \lambda_t w_\lambda + \delta_t w_\delta \tag{1}$$

Given total node count, l, target cell population, n, and the number of nodes eligible for *ACTIVE* duty, m, we can obtain the average rate of energy consumption for a network cell in the steady state. From section 3.2, $\delta = n$, $\lambda = m - n$, and $\gamma = l - m$.

$$w_{cell} = (l - m)w_\gamma + (m - n)w_\lambda + nw_\delta \tag{2}$$

Using typical sensornet mote platforms [8], *SEARCHING* and *JOINING* nodes in Λ have similar energy consumption to *ACTIVE* nodes in Δ, and *SUSPENDED* nodes in Λ have similar energy consumption to *INACTIVE* nodes in Γ. This implies equation 3.

$$w_{cell} = (l - m + (1 - p_\omega)(m - n))w_\gamma + (n + p_\omega(m - n))w_\delta \tag{3}$$

8 Results

We now present experimental results obtained by measuring ADCP performance in simulated sensornets.

8.1 Measuring effectiveness

We define the metrics $R_1 - R_3$ to measure the effectiveness of ADCP. As ADCP has multiple objectives, it follows that we require multiple metrics with which to measure success against these objectives as runtime t increases in the interval $[0, \infty)$. Metrics are sampled once per system epoch during this period.

R_1: The size δ of the *active set*, Δ, population at some time t. This is the *actual* value of δ available to an omniscient observer, not the *estimate* used by cell members in probability calculations. Measured in *nodes*. Defined in the range $[0, l]$ where l is the total number of nodes in a cell. The ideal value of $R_1 = n$.

R_2**:** The proportion of total network runtime during which nodes are *SUSPENDED*, taken as the average across all l nodes, during which nodes can switch off energy-hungry wireless communications modules. Unitless. Defined in the range $[0, 1]$. The ideal value of $R_2 = 1$.

R_3**:** The standard deviation of proportion of time for which nodes are assigned to the *active set*, Δ, across all non-*INACTIVE* nodes in the set $(\Lambda \cup \Delta)$. Unitless. Defined in the range $[0, 1]$. The ideal value of $R_3 = 0$.

8.2 Cold start

Initially all nodes are *SUSPENDED*, with $R_1 = 0$ defining a *maximally underpopulated cell*. As time progresses, we expect that $R_1 \to n$ as nodes independently transition from Λ to Δ. We set *node activation coefficient* $\pi_\psi = 1$, and *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave Δ. We set *voluntary suspension probability* $p_\tau = 0$ to avoid confounding population stabilisation and fair duty allocation effects.

We first consider R_1 as a function of time, t, as measured in system epochs of length e time units. We set $l = 20$ and $n = 10$; despite the node surplus, active cell population should reach $\delta = n$ rather than $\delta = l$

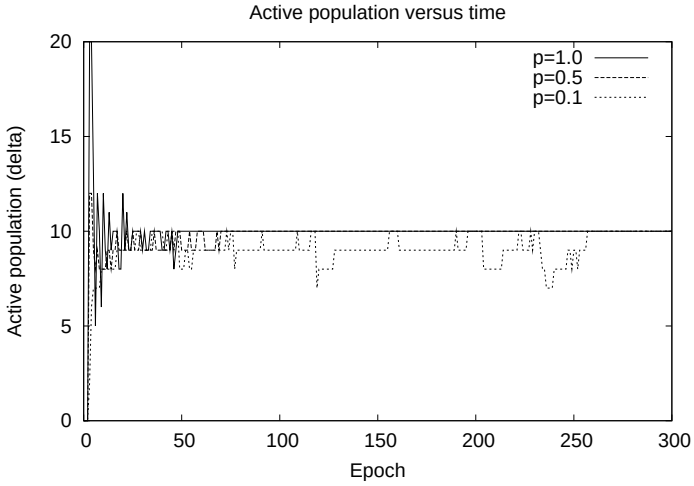

Figure 2: R_1 over time for values of p_ω

Figure 2 illustrates the relationships between R_1 and time for different values of p_ω. The *suspended-searching probability* p_ω varies in the range $[0, 1]$, with one value of p_ω per trace. Eventually $R_1 = n$ for all $p_\omega > 0$, with higher p_ω reaching this more quickly. It is desirable to set p_ω as high as possible while still conforming to cell energy usage requirements.

Consider the relationship between p_ω and the critical time t_c required to reach the stable condition $R_1 = n$. Figure 3 plots p_ω against t_c, measured in epochs. Examining the smoothed trend, higher values of p_ω yield lower values of t_c, indicating that higher p_ω is desirable for rapid stabilisation. Examining the unsmoothed raw datapoints, the relationship is less simple. Where t_c appears particularly large for p_ω, often $R_1 \approx n$, cycling between n and either $R_1 = n - 1$ or $R_1 = n + 1$ for numerous epochs before settling on $R_1 = n$.

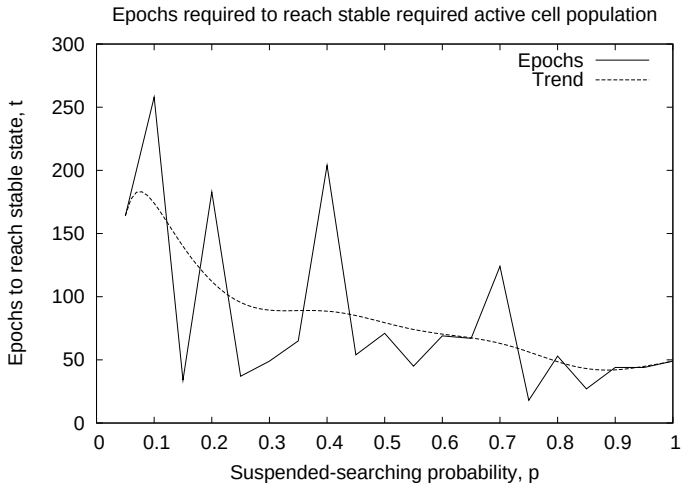

Figure 3: Epochs to $R_1 = n$ versus p_ω

ADCP should function correctly where l varies. It should maintain $R_1 = n$ if $l \geq n$, and maintain $R_1 = l$ where $l < n$. Figure 4 illustrates the time measured in epochs to reach the stable target population for each value of l on a logarithmic scale. We set $p_\omega = 0.5$ as the central value of the defined range $[0, 1]$, and varied l in the interval $[1, 100]$ to span orders of magnitude greater, and lesser, than the critical value $l = n$.

Firstly, consider the plot for $l < 10$ in figure 4. Relatively few epochs pass until reaching the $R_1 = l$ stable condition for $l < n$. The cell is always *underpopulated*, so any *SEARCHING* node can become *ACTIVE*. As $l \to n$ a greater number of epochs are generally required as more nodes must probabilistically decide to join the *active set*, Δ.

Secondly, consider the plot for $l \geq 10$ in figure 4. The number of epochs until reaching the $R_1 = n$ stable condition for $l \geq n$ is generally higher than for $l < n$, because the cell may be *overpopulated*, *underpopulated* or *correctly populated*. The algorithm cannot simply allow all *SEARCHING* nodes to become *ACTIVE*; unless restricted, R_1 would grow until reaching the $R_1 = l$ upper bound, overshooting $R_1 = n$.

For $l \geq 10$, the "Epochs" plot is jagged, indicating it is non-trivial to predict t_c for a given l. This is a consequence of initial LISP stabilisation.

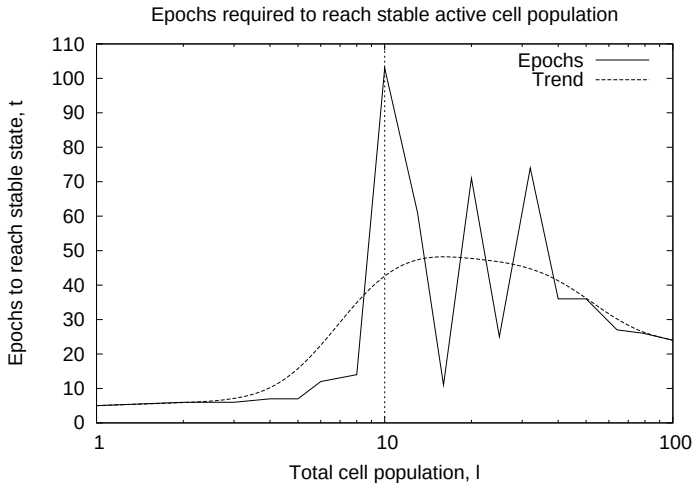

Figure 4: Epochs to $R_1 = n$ versus l

However, the trend is for t_c to decline with increasing l; the target active population $\delta = n$ is reached more quickly with larger total cell populations, which have a greater number of suitable candidate nodes. ADCP functions more effectively as l becomes larger. This scalability property is particularly useful in very large sensornets.

8.3 Replacing a failed node

We consider the removal of *ACTIVE* node from a cell in which the $R_1 = n$ condition already holds. The cell has l nodes, of which $n < l$ should be in the *active set*, Δ, at any given time. The cell is stable at the outset, and remains so until a single node *ACTIVE* is selected at random and becomes *INACTIVE* after a delay of t_b. We observe R_1 as a function of time until ADCP replaces the removed node, and the $R_1 = n$ condition is later regained at some time t_c.

The cell has total population $l = 20$ and target active population $n = 10$. We set *node activation coefficient* $\pi_\psi = 1$, and *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave the *active set*, Δ. We set *voluntary suspension probability* $p_\tau = 0$ to avoid confounding population stabilisation and fair duty allocation effects. We set *suspended searching probability* $p_\omega = 0.5$ as the central value of the defined range $[0, 1]$.

The cell is stable until epoch $t_a = 50$ at which point an *ACTIVE* node is removed. *SEARCHING* members of the *reserve set*, Λ, implicitly detect the missing node as they estimate $\delta = 9$ rather than $\delta = 10$. During the subsequent 49 epochs, R_1 varies between 9 and 10 as ADCP replaces the missing node; this variation is explained by inaccuracies in subpopulation

estimation. Finally, the cell reattains a stable active population of $R_1 = n = 10$ at time $t_c = 50 + 49 = 99$ epochs.

8.4 Removing a surplus node

We consider the addition of an extra *ACTIVE* node to a cell in which the $R_1 = n$ condition already holds, reusing the experimental configuration from section 8.3. If this extra node was non-*ACTIVE* it would implicitly be added to the set of spares available to replace failed nodes. However, as it is *ACTIVE*, then $R_1 = n + 1$ and thus ADCP must transition one node from the *active set*, Δ, to the *reserve set*, Λ.

The cell is stable until epoch $t_a = 50$ at which point a new *ACTIVE* node is added. Members of the *active set*, Δ, implicitly detect the surplus node as they estimate $\delta = 11$ rather than $\delta = 10$. During the subsequent 2 epochs, all 11 members of Δ independently decide whether to transition to the *reserve set*, Λ. The cell reattains a stable active population of $\delta = n = 10$ at time $t_c = 50 + 2 = 52$ epochs.

t_c is shorter for removal of a surplus node than replacement of a failed node as members of Δ are constantly estimating cell population sizes, and can react immediately to a perceived surplus. By contrast, members of Λ only estimate population sizes when *SEARCHING*, and their reaction is non-immediate; they must follow the *SEARCHING*→*JOINING*→*ACTIVE* sequence, spanning a duration of $1 - 2$ epochs.

8.5 Proportion of time in ADCP states

Higher values of the *suspended-searching probability*, p_ω, allow cells to attain the desired $\delta = n$ condition more quickly. However, increasing p_ω induces nodes in Λ to spend more time in *SEARCHING*, thus consuming more energy. These competing responsiveness and efficiency demands must be balanced by setting p_ω.

R_2 measures the potential to save energy by placing unused nodes into low-energy sleep states. The cell has target active population $n = 10$ and total population $l = 100$. We expect the *reserve set*, Λ, to contain 90 nodes. All nodes begin in the *SUSPENDED* state.

We set *node activation coefficient* $\pi_\psi = 1$, and *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave the *active set*, Δ. We set *voluntary suspension probability* $p_\tau = 0$ to avoid confounding population stabilisation and fair duty allocation effects. We set *suspended searching probability* $p_\omega = 0.1$ as this is within the defined range $[0, 1]$ and allows nodes in Λ to spend the majority of time in the *SUSPENDED* state.

Figure 5 shows the average proportion of time spent by nodes during the first 10 epochs as an area plot, with the *ACTIVE*, *JOINING*, *SUSPENDED*,

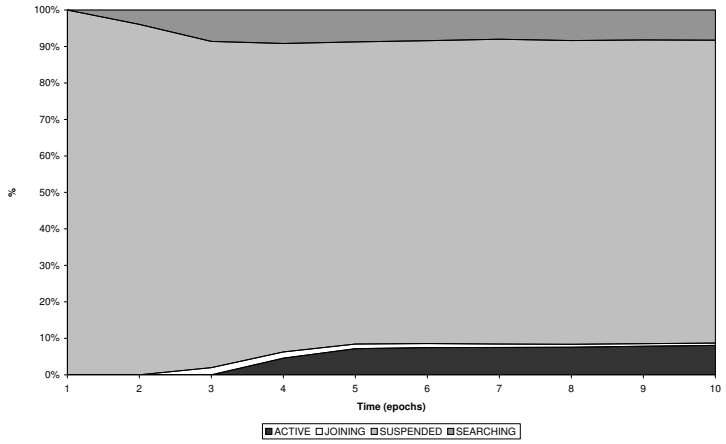

Figure 5: Time in ADCP states: first 10 epochs

and *SEARCHING* states in ascending vertical order. The proportions of cumulative time spent in each state quickly approach their stable values. The *SUSPENDED* state dominates, which is the desired outcome. The time allocated to *ACTIVE* and *SEARCHING* is significant, though much smaller. The time allocated to *JOINING* is initially small and quickly becomes insignificant.

Compare the numerical results from this experiment, continued to 500 epochs to allow further stabilisation, against values predicted by section 5. Results are given in table 2 to 4 decimal places. Experimental and predicted values are identical within $\pm 0.01e$.

State	Experiment	Predicted	Error (absolute)
ACTIVE	0.0996	0.1000	0.0004
JOINING	0.0001	0.0001	0.0000
SUSPENDED	0.8179	0.8100	0.0079
SEARCHING	0.0824	0.0900	0.0076

Table 2: Proportion of time: first 500 epochs

8.6 Fairness and equality

This experiment considers a cell of total population $l = 20$ and target active population $n = 10$. Nodes are ideally *ACTIVE* for 50% of network runtime such that $R_3 = 0$. We set the *node activation coefficient* $\pi_\psi = 1$, and the *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave the *active set*, Δ. We set the *suspended searching probability*, $p_\omega = 0.5$, as this is the central value of the defined range $[0, 1]$. The *voluntary suspension probability*, p_τ, is varied in $[0, 1]$ and the resulting value of R_3 measured.

Consider the value of R_3 as a function of time, t, where $p_\tau > 0$. Shortly

after the experiment begins, the $\delta = n$ goal is attained as n of l nodes are selected as *ACTIVE*, and the remaining $l - n$ are not. At this stage, the n selected nodes have been *ACTIVE* for a large proportion of time, and the $l - n$ other nodes for only a small proportion; the standard deviation (R_3) of these *ACTIVE* proportions will be relatively large, of the same order of magnitude as the mean.

If $p_\tau > 0$, as $t \to \infty$ nodes will occasionally be shuffled between Δ and Λ. All l nodes are eventually selected as *ACTIVE* for approximately equal proportions of total time, so R_3 eventually becomes small.

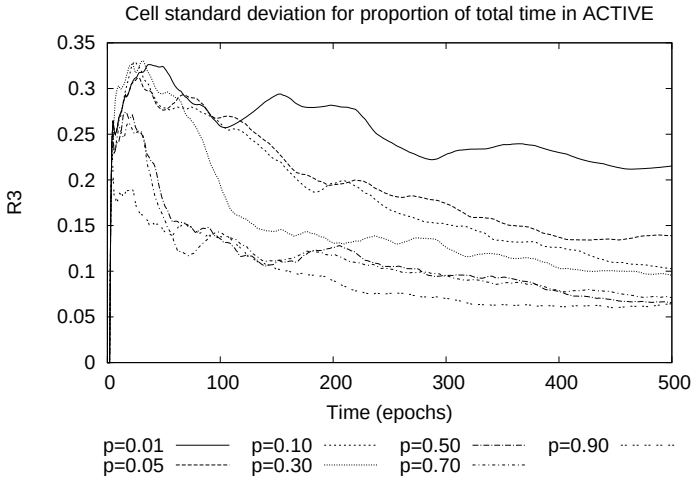

Figure 6: Std. dev. of *ACTIVE* time proportion

Figure 6 illustrates the decline of R_3 as a function of time, t, for differing values of p_τ. As R_3 declines, this represents an increasingly fair distribution of burden between all l nodes. Higher values of p_τ lead to a sharper decline in R_3 and hence a faster equalisation of burden across the cell population. For long-running networks low p_τ values minimise churn while achieving equal burden sharing in the long term.

Figure 7 illustrates R_3 for differing values of p_τ after a fixed duration from network startup of t epochs, each of 10 seconds. Sampling R_3 at time t measures the overall success of ADCP at fairly distributing burden for a sensornet of expected lifetime t. Over time $R_3 \to 0$ as $t \to \infty$, but in sensornets with finite lifetime the $R_3 \approx 0$ condition may not be reached.

Values of t were selected spanning 5 orders of magnitude from $t = 5 \times 10^2$ to $t = 5 \times 10^6$ epochs. As $t \to \infty$ the selected value of p_τ becomes less important; there is less variance between R_3 values, and all approach the ideal $R_3 = 0$. Over long periods the transition rate controlled by p_τ becomes irrelevant, provided it is non-zero, as all nodes will eventually spend approximately equal time in Δ.

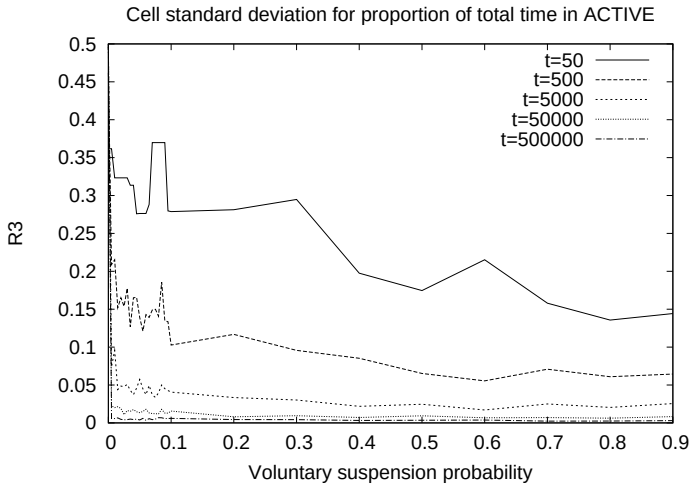

Figure 7: *ACTIVE* time std. dev. over timescales

9 Conclusions

This paper defined and evaluates the *Active Duty Control Protocol* (ADCP), which manages active node populations within sensornet cells. Participating nodes are allocated between an active pool of fixed size, and a reserve pool containing the remainder. Nodes are moved between pools for wear balancing, and when necessary to correct the active population size.

Probabilistic methods achieve rapid convergence on the target population size with low overheads in computation, storage and energy. Empirical evaluation demonstrates that ADCP maintains active node populations efficiently and reliably in the long term, with fair wear balancing across all nodes. Energy consumption is minimised by maximising the proportion of time nodes can disable energy-hungry subsystems.

Acknowledgements

This work is partially supported by an EPSRC grant, and by BAE Systems plc under the grant *Hierarchical System Management for Integrated Modular Systems.*

References

[1] G. Box, J. Hunter, and W. Hunter. *Statistics for Experimenters.* Wiley, Hoboken, NJ, 2nd edition, 2005.

[2] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proc. International Conference*

on Acoustics, Speech, and Signal Processing, pages 2033–2036, May 2001.

[3] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proc. 5th International Conference on Mobile Computing and Networking*, pages 263–270, 17-19 August 1999.

[4] C. Hua and T. Yum. Asynchronous random sleeping for sensor networks. *ACM Transactions on Sensor Networks*, 3(3):710–714, 2007.

[5] Y. Pan and X. Lu. Energy-efficient lifetime maximization and sleeping scheduling supporting data fusion and QoS in Multi-Sensornet. *Signal Processing*, 87(12):2949–2964, 2007.

[6] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S. Iyengar. Random asynchronous wakeup protocol for sensor networks. In *Proc. 1st International Conference on Broadband Networks*, pages 710–717, October 2004.

[7] A. Patel, J. Degesys, and R. Nagpal. Desynchronization: The theory of self-organizing algorithms for round-robin scheduling. In *Proc. 1st International Conference on Self-Adaptive and Self-Organizing Systems*, pages 87–96, 9-11 July 2007.

[8] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[9] J. Tate and I. Bate. Energy efficient duty allocation protocols for wireless sensor networks. In *Proc. 14th IEEE Conference on Engineering of Complex Computer Systems*, pages 58–67, 2-4 June 2009.

[10] J. Tate and I. Bate. An improved lightweight synchronisation primitive for sensornets. In *Proc. 6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 448–457, 12-15 October 2009.

[11] K. Wu, Y. Gao, F. Li, and Y. Xiao. Lightweight deployment-aware scheduling for wireless sensor networks. *Mobile Networks and Applications*, 10(6):837–852, 2005.

[12] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. 23rd International Conference on Distributed Computing Systems*, pages 28–37, 19-22 May 2003.