Maximizing Energy Efficiency for Convergecast via Joint Duty Cycle and Route Optimization

Wenjie Zeng, Anish Arora, and Ness Shroff
Department of Computer Science and Engineering, Ohio State University, Columbus, Ohio 43210
Email: {zengw, anish, shroff}@cse.ohio-state.edu

***Abstract*—The energy efficiency of the widely used convergecast pattern depends substantially on the choice of medium access control (MAC) and routing protocol. In this paper, we formalize the maximization of convergecast energy efficiency with respect to its MAC and routing as a resource constrained optimization problem. We then analytically show that this maximization problem is linear in the context of two prototypical MACs — a locally synchronized wakeup (as in S-MAC) and a locally staggered wakeup MAC (as in O-MAC)— assuming low, uniform traffic that is delivered reliably and without interference. With this insight, we present a centralized algorithm, MeeCast, that solves the optimization problem utilizing linear programming techniques. We also design a distributed version of MeeCast, for the case where the traffic is ultra-low, and prove that it achieves optimality as well as fast convergence time. Notably, this version is self-stabilizing, so it autonomically handles changes in traffic load, network topology, loss of coordination and state corruption. In comparison with a state-of-the-art convergecast protocol, Dozer, MeeCast achieves better energy efficiency and application lifetime in the context of S-MAC and identical energy efficiency but better application lifetime in the context of O-MAC.**

I. Introduction

Monitoring is the predominant use case for Wireless Sensor Networks (WSNs) today, cf. [1], [2], [3], [4]. Thus convergecast, wherein data from children nodes is collected by parent nodes and progressively forwarded towards one or more base stations, is a common case operation and its energy efficiency is of systemic importance. By energy efficiency, we informally mean the amount of data collected by the base station per unit of energy consumed by the whole network.

In general, optimization for energy efficiency can be performed at multiple layers ranging from application level data aggregation, to transmission power control and path selection, to traffic and wakeup scheduling, down to physical layer channel selection schemes. Layers are often interdependent. Recent work on cross layer optimization ([5], [6]) has shown that optimization within a single layer usually fails to achieve optimality because it does not address layer interdependencies. We are therefore motivated to consider optimizing convergecast energy efficiency in a cross-layer fashion.

It is by now well understood that radio receiver power consumption is a dominant component of the energy equation in low-power WSNs [7]. In other words, minimization of radio idle listening, overhearing of traffic intended for others, and interference has substantial efficiency impact from a systems analysis viewpoint. Our problem focus therefore includes control of traffic scheduling, which determines when a node should send or receive, as well as wakeup scheduling, which determines how long it should be active.

Both traffic and wakeup scheduling are functions of the MAC. In conventional network architectures, the control of the former is usually internal to the MAC, whereas that of the latter is usually exposed so that higher layers can choose the overall duty cycle. By duty cycle, we mean the percentage of time a node's radio is activated by its MAC; the duty cycle can be further decomposed as reception (respectively, transmission) duty cycle, which determines the percentage of time a node receives (respectively, transmits). Exposing duty cycle control on a per node basis to the routing layer is particularly relevant for energy efficiency of multi-hop convergecast, since convergecast load varies from node to node and hop to hop. A node operating a node at a duty cycle higher than necessary will waste energy by idle listening, whereas operating at an insufficient duty cycle will fail to satisfy its local communication load, which depends on its local data generation rate and the routing layer induced traffic.

Conversely, the routes chosen for convergecast depend on MAC layer scheduling. Routing is typically based on metrics such as Packet Reception Rate (PRR) and latency, which in turn depend on MAC scheduling, especially when the duty cycle is optimized to just accommodate the current traffic: A lower than desirable duty cycle results in low reliability and high latency; the same is true for a duty cycle allocation to nodes in a neighborhood that is higher than the channel capacity in that neighborhood can accommodate. Moreover, routing needs to account for duty cycle constraints of nodes. For instance, Zhao et al [8] have shown that although link quality is time variant, good links are relatively stable over relatively long time windows, and thus nodes on the reliable paths are likely to remain as preferred next hops for multiple children and thereby drain their battery much faster. Selecting only nodes with the best links reduces application lifetime, so it is potentially preferable to choose paths over less preferable hops in order to accommodate node energy constraints.

In sum, a scheme that optimizes efficiency within one layer only will fail to address this cyclic dependency between the MAC and routing layers. Our problem focus thus accommodates joint routing and duty cycle control.

Contributions of the paper. In this paper, we formalize the maximization of energy efficiency with regard to MAC scheduling and routing as a resource constrained optimization problem. We then analytically show that this maximization problem is linear in the context of two prototypical MACs — a locally synchronized wakeup (as in S-MAC) and a locally staggered wakeup MAC (as in O-MAC)— assuming uniform, low traffic that is delivered reliably and without interference. Given the linearity, there is an efficient centralized algorithm, MeeCast, for solving the optimization problem even at a computationally-constrained base station. In comparison with a state-of-the-art convergecast protocol, Dozer, MeeCast achieves better energy efficiency and application lifetime in the context of S-MAC and identical energy efficiency but better application lifetime in the context of O-MAC.

Our assumptions warrant some discussion. Low traffic is the norm for WSNs, as monitoring data rates are invariably well below the network capacity (i.e., [2], [3], [4]). Reliable delivery and no interference may be assumed only because joint control of the MAC and routing can avoid collisions and select desirable links when the traffic is low. (It should be noted that works such as ([9], [10]) that take interference into account during optimization, coupled with power control and scheduling, usually show non-convex constraints over link rates, which makes efficient solutions difficult to achieve. Efficient solutions for optimization problems involving scheduling have typically involved approximation algorithms or TDMA MACs.)

Our assumption of uniform traffic is however restrictive in that it focuses on use cases where the data generation and flow is periodic in some fashion for nontrivial lengths of time. For the case where the traffic changes on a slow time scale, MeeCast can be run when a change is detected. We also present the alternative of using a distributed version of MeeCast that is self-stabilizing. By self-stabilizing, we mean that every computation of the network, upon starting from an arbitrary (but, ultra-low) load and network state, always converges within finite time to a state where the network is operating optimum duty cycle and routes. This property also provides autonomic recovery to diverse faults, especially node failure which is typically non-trivial for the commodity nodes used in conventional, as well topology, loss of coordination and state corruption.

Organization of the paper. In Section II, we discuss preliminaries and formulate the energy efficiency problem as a constrained flow optimization problem. In Section III, we analyze the communication capacity of nodes under O-MAC and S-MAC. We describe our centralized and distributed solutions for maximizing energy efficiency via simultaneous routing and duty cycle optimization (our proof of self-stabilization is relegated to a technical report [11] for reasons of space.) We validate performance via simulations in Section V, discuss related work in Section VI, and make concluding remarks in Section VII.

II. Problem Statement

We begin with the system model, formalize the notion of energy efficiency of convergecast, and then state the energy efficiency problem.

A. System Model

A wireless sensor network is represented by a directed graph $G = (V, L)$, where V is a non-empty set of wireless nodes, one of which is distinguished as the base station bs, and L is a set of half-duplex, reliable links L. Let i, j range over $1..|V|$ and l over $1..|L|$. (For convenience, we henceforth abbreviate $|V|$ as V and $|L|$ as L.)

Node v_j can receive messages from v_i if and only if $(v_i, v_j) \in L$. Let us define node incidence matrix A, where $A \in \mathbb{Z}^{V\times L}$, as:

$$A_{il} = \begin{cases} 1 & \text{, if } v_i \text{ is the start node of } l \\ -1 & \text{, if } v_i \text{ is the end node of } l \\ 0 & \text{, otherwise} \end{cases} .$$

Let us also define matrix A^+, $A^+ \in \mathbb{Z}_+^{V\times L}$, as $A^+_{il} = 1$ if $A_{il} > 0$ and $A^+_{il} = 0$ otherwise. And matrix A^-, where $A^- = A^+ - A$.

Let $x \in \mathbb{R}_+^L$ be the rate vector in which x_l is the data rate over link l and $s \in \mathbb{R}_+^V$ be the source rate vector in which s_i is the source rate of v_i. For $l = (i, j)$, we will use x_l and x_{ij} interchangeably. We assume that all source rates are constant albeit not identical. For each link $l \in L$ and $i \in \{V - \{bs\}\}$, $x_l \geq 0$ and $s_i \geq 0$. Introduce an extra vector s' in which $s'_i = s_i$ if $i \neq bs$ and $s'_i = 0$ otherwise.

Denote $\mathcal{I}_i$ and $\mathcal{O}_i$ as the sum of incoming data rates to and outgoing data rates from v_i respectively, where

$$\mathcal{I}_i = (A^- x)_i \quad , \quad \mathcal{O}_i = (A^+ x)_i \quad .$$

In light of flow conservation law, we have

$$\mathcal{I}_i + s_i = \mathcal{O}_i \quad , \quad s_{bs} = -\sum_{i\in\{V-bs\}} s_i .$$

B. Energy Efficiency

Let T be the running time of the application and d_i the percentage of active time of the radio at v_i. Assume that the energy consumption rate for staying active, be it sending or receiving, is a constant, the energy E_i spent by v_i during T is then measured by $d_i T$. Each node v_i has a duty cycle upper bound D_i, $D_i \in (0, 1]$. D_i is an application specific parameter which is useful for multiple purposes, such as to guarantee the lifetime of the application or to ensure fairness.

In generic network contexts, energy efficiency is defined to be the reciprocal of the energy consumed by all participating nodes in the network, denoted as E_{abs} in (1). Given that the merit for a monitoring applications is its ability to collect data from sensor nodes, we propose here a convergecast specific definition of energy efficiency, denoted as E_c in (1), to be the ratio of the total number of useful data packets received by base station to the total energy spent.

$$E_{abs} = \frac{1}{\sum\limits_{i\in V} d_i T} \quad , \quad E_c = \frac{\mathcal{I}_{bs} T}{\sum\limits_{i\in V} d_i \times T} = \frac{\mathcal{I}_{bs}}{\sum\limits_{i\in V} d_i} . \qquad (1)$$

E_c captures the cost effectiveness of the system whereas a inefficient system collecting zero data at the base station can still have a high E_{abs} as long as the sum of the duty cycles is low.

Time synchronization and other control messages are common radio energy overheads that depend on the actual implementation of the protocols. We eschew further consideration of these overheads in this work.

C. The MaxEE-Convergecast Problem

The energy efficiency optimization is formalized as a constrained network flow optimization problem. Given A, s, and D as constant inputs, we formulate the objective function $E_c(x, d)$ as follows:

$$(\texttt{P}) \quad E_c(x, d) = \mathcal{I}_{bs} / \sum_{i \in V} d_i \ ,$$

in which vector d controls the duty cycle of nodes and vector x defines the routing and the amount of traffic per link.

We define $\mathcal{R}_i(r_i, x)$ as the *receiving capacity* of v_i, i.e., the maximum data rate v_i can successfully receive when it spends r_i percentage of time in receiving under traffic x. Similarly, we define $\mathcal{F}_i(t_i, x)$ as v_i's *forwarding capacity* with a transmission duty cycle t_i. The *capacity* of v_i, denoted as $\theta_i(d_i, x)$, is defined as

$$\theta_i(d_i, x) = \min_{r_i + t_i = d_i} \{\mathcal{R}_i(r_i, x) + s'_i \ , \ \mathcal{F}_i(t_i, x)\} \quad . \tag{2}$$

Function θ depends on factors such as the underlying MAC protocol, interference range, etc. Our optimization problem is then expressed as:

Maximize $E_c(x, d)$

Subject to

$$\begin{aligned} &1. \quad Ax = s \\ &2. \quad 0 \le x_l, \forall\, l \in L \\ &3. \quad 0 \le d_i \le D_i, \forall\, i \in V \\ &4. \quad 0 \le \theta_i(x, d) - (\mathcal{I}_i + s'_i), \forall\, i \in V \end{aligned} \tag{3}$$

The first and second constraint ensure the flow conservation law and valid link rates, respectively. The third constraint imposes nodal duty cycle constraint. The last constraint ensures that every node runs at a sufficient duty cycle such that its capacity is greater than the communication demand. We define the set of feasible solutions $\Omega(A, s, D)$, henceforth abbreviated as Ω, for problem $(\texttt{P})$ with parameters (A, s, D) to be:

$$\Omega(A, s, D) = \{(\hat{x}, \hat{d}) : (\hat{x}, \hat{d}) \text{ satisfies all constraints in (3)}\} \ .$$

For any feasible solution $(x, d) \in \Omega$, $\mathcal{I}_{bs} = \sum_{i \in V} s'_i$ must hold. Since the optimal solution must be feasible, we can rewrite the problem (P) as:

$$(\texttt{P1}) \quad \text{Minimize} \quad Q_c(x, d) = \sum_{i \in V} d_i / \sum_{i \in V} s'_i \ ,$$

subject to the same set of constraints in (3). Clearly Problem $(\texttt{P1})$ and $(\texttt{P})$ have the same solution because the objectives are reciprocal and their constraints are the same.

Since $\sum_{i \in V} s'_i$ is assumed to be constant, in order to minimize Q_c we essentially need to minimize $\sum_{i \in V} d_i$ given the constraints in (3). In the next section, we show that $\theta_i(d_i, x)$ is also linear in x and d for a number of representative MAC protocols when they can be operated to achieve interference-free communication scheduling. This enables us to solve our optimization problem using linear programming techniques.

III. Capacity under Extant MACs

In this section, we first analyze capacity function θ under a theoretical TDMA in which all communications are perfectly scheduled. This TDMA idealizes the class of MAC protocols in which nodes are aware of their neighbors' schedules and can control their duty cycle independently, such as O-MAC [7]. We show that, under low traffic, O-MAC approximates the interference-free scheduling of the ideal TDMA and the capacity constraint in (3) is then a linear in x and d under O-MAC.

Next we analyze S-MAC [12]. This represents a different class of MAC protocols where the nodes cannot control duty cycle independently. We show that, under the assumption that S-MAC can achieve interference-free transmission, problem $(\texttt{P1})$ is still in the linear space.

We denote N_i as the 1-hop neighborhood and η_i as the interference range of v_i. Both N_i and η_i are determined by A. η_i also depends on other constants such as the relative locations of nodes and the transmission power. Since A is constant, both N_i and η_i can be regarded as constants.

A. TDMA Protocols

Under ideal TDMA, nodes are perfectly synchronized and an oracle is used to schedule so that there is no transmission collision, idle listening, or overhearing. More formally, let t_s be the time it takes for a node to send or receive a packet. Define $\alpha_i(j, t)$ as the send schedule for v_i where $\alpha_i(j, t)$ equals 1 if v_i is scheduled to send to v_j during $(t, t + t_s)$ and 0 otherwise. Similarly, define $\beta_i(j, t)$ as the receive schedule. When $\alpha_i(j, t) = \beta_j(i, t) = 1$, up to one packet can be successfully sent from v_i to v_j. The following properties are satisfied by the ideal TDMA, for any $i, j, m, n \in V, j \ne i, n \ne i, m \ne n, a \in \eta_i, a \ne i \ , t, T \in \mathbb{Z}_+$:

$$\begin{aligned} &1. \quad \alpha_i(j, t) = 0 \ \vee \ (\alpha_i(j, t) + \beta_j(i, t) = 2) \\ &2. \quad \alpha_i(m, t) \cdot \alpha_i(n, t) = 0 \wedge \alpha_i(j, t) \cdot \beta_i(n, t) = 0 \\ &3. \quad (\alpha_i(j, t) \cdot \beta_a(n, t) = 0) \wedge (\beta_i(j, t) \cdot \alpha_a(n, t) = 0) \\ &4. \quad \sum_{\tau \le T} \alpha_i(j, \tau) = \sum_{\tau \le T} \beta_j(i, \tau) = \lfloor x_{ij} T \rfloor \ , \end{aligned} \tag{4}$$

Property 1 guarantees that when v_i communicate with v_j, v_j is scheduled accordingly. Property 2 states that a sender does not send to more than one node and an active node can either receive or send at a time. The third property specifies that senders within the interference range of a node do not transmit at the same slot and receivers within the interference range of the same sender do not receive at the same slot (to avoid over-hearing). The last property ensures that for any link

(i,j), the sending (receiving) slots assigned to v_i (v_j) during a period of T is equal to $x_{ij}T$.

Capacity θ_i is affected by both duty cycle d_i and the interfering traffic. Under ideal TDMA, since every packet transmission and reception is well scheduled, there is no interference, idle listening, or overhearing. The receiving capacity and sending capacity is thus computed as:

$$\mathcal{R}_i = r_i/t_s \quad \text{and} \quad \mathcal{F}_i = (d_i - r_i)/t_s \ .$$

Theorem 1. *Under ideal TDMA, the capacity $\theta_i(d_i,x)$ of node i is maximized to $(d_i+s'_i t_s)/2t_s$ when it spends $(d_i-s'_i t_s)/2$ percentage of time in receiving. The optimization problem* (P1) *is thus linear.*

Proof: Let $r_i^* = (d_i - s'_i t_s)/2$. Let $r_i \neq r_i^*$ be the percentage of time node i receives. (i) If $r_i > r_i^*$, we have $\mathcal{F}_i = (d_i - r_i)/t_s < \mathcal{R}_i + s'_i$, then $\theta_i(d_i,x) = \mathcal{F}_i < (d_i - r_i^*)/t_s = (d_i+s'_i t_s)/2t_s$; (ii) If $r_i < r_i^*$, then $\mathcal{R}_i + s'_i < \mathcal{F}_i$ and $\theta_i(d_i,x) = \mathcal{R}_i + s'_i < r_i^*/t_s + s'_i = (d_i+s'_i t_s)/2t_s$. Therefore, the capacity of node i is maximized to $(d_i + s'_i t_s)/2t_s$ when it spends $(d_i - s'_i t_s)/2$ percentage of time in receiving.

Replacing $\theta_i(d_i,x)$ with $(d_i + s'_i t_s)/2t_s$ in the capacity constraint in (3), we transform it into

$$0 \leq (d_i + s'_i t_s)/2t_s - (\mathcal{I}_i + s'_i) \Longleftrightarrow (2\mathcal{I}_i + s'_i)t_s \leq d_i \ . \quad (5)$$

Given that s'_i is a constant, we see that the capacity constraint is linear in d and x, and thus problem (P1) is linear. ■

Right after the beginning of data collection, the effective data rate over some link (i,j) might be less than x_{ij} because the traffic routed through it might not have arrived, resulting in idling listening slots at the receiver. Once all flows routed through link (i,j) reach v_i, because of the alignment between the communication schedule and the uniform traffic, the effective data rate will be steady and equals to x_{ij}. Since the distance between any node and the base station is bounded by V, the number of such idle listening slots is also bounded. As a result, the energy overhead due to such initial idle listening is bounded by a constant and is negligible for large T.

Of course, ideal TDMA assumes full information about the network and the traffic, which is usually not available. O-MAC provides a distributed implementation of ideal TDMA that approximates its properties under low traffic. In O-MAC, the global time is split into frames and each frame is further split into T_p slots, each of which is t_s in length. Time-synchronized receivers are scheduled either globally and deterministically or, more efficiently, pseudo-randomly to wake up and receive in each frame. In the pseudo-random scheme, each node locally selects its receiving schedule. Under low traffic, the authors in [7] show that the probability that two receiver schedules overlap is negligible. A node's receiving frame schedule is broadcast to its neighbors which then contend for transmission during the receiving slots. Each receiving frame schedule contains a fixed number of receiving slots that is determined by the node's reception duty cycle. Therefore, O-MAC naturally supports the first three properties in (4) except for interference-free transmission. However, for any given x, it is in principle feasible for a receiver to attach a non-overlapping sending schedule for the contending senders along with its receiving schedule. Each child v_j of v_i would be assigned $T_p x_{ij} t_s$ sending slots per frame in order to align with the traffic x_{ij}. As a result, O-MAC satisfies the properties of (4) except that the alignment between the communication schedule and the traffic flow would be on a per-frame basis, which may require a longer queue at each node.

B. S-MAC

In contrast, S-MAC organizes nodes in clusters, and all nodes of each cluster are scheduled to wake up at the beginning of every frame. Nodes try to join an existing cluster before they attempt to create their own cluster. Nodes that follow more than one wakeup schedule are called the *border nodes*. For ease of analysis, we assume that the whole network is within one synchronized cluster and so all nodes share the same wake-up schedule. S-MAC utilizes RTS-CTS messages to schedule contending senders which makes interference negligible under low traffic. In other words, nodes pay the price of control messages for achieving interference-free communication.

Since every data packet requires an RTS-CTS exchange, assuming that sending RTS or CTS, including the initial back off, takes one slot each, then every packet transmission takes three slots. RTS-CTS messages preclude any node within the neighborhood of the sender or the receiver to send during the same period. As a result, for a data rate x_{ij}, it takes $3x_{ij}T_p t_s$ slots per frame in the shared slots of v_i and v_j's neighborhoods to successfully transmit it. Since duration information is embedded in all RTS-CTS messages, we assume that packet transmissions are back-to-back such that the transmission of one data packet is immediately followed by another round RTS-CTS-DATA transmission in the next three slots.

Because of the shared wakeup period, the last constraint in (3) can no longer be defined on a nodal basis. In the worst case, each node within N_i sends and receives at different slots, during all of which v_i has to be awake. Thus, given the neighborhoods's total traffic is equal to $3\sum_{j\in N_i}(2\mathcal{I}_j + s'_j)$, v_i has to be awake for $3\sum_{j\in N_i}(2\mathcal{I}_j + s'_j)t_s$ to accommodate it in the worst case.

We modify the constraints in (3) to reflect the unique shared wakeup period feature of S-MAC. The new constraints are shown as follows:

$$\begin{array}{ll} 1. & Ax = s \\ 2. & 0 \leq x_l, \text{for } \forall\, l \in L \\ 3. & 0 \leq d_i \leq D_i, d_i = d_j, \text{ for } \forall\, i,j \in V \\ 4. & d_i \geq 3\displaystyle\sum_{j\in N_i}(2\mathcal{I}_j + s'_j)t_s, \forall\, i \in V\ . \end{array} \quad (6)$$

Based on this analysis, when S-MAC is used and all nodes are within the same synchronized cluster, the optimization problem of minimizing $Q_c(x,d)$ subject to constraints defined in (6) is a linear optimization problem.

IV. Solution

We now present MeeCast that solve the optimization problem for maximal energy efficiency convergecast. Section IV-A presents centralized version MeeCast that requires global information at the base station. With an additional assumption about traffic being low, Section IV-A first proves the optimality condition under the assumption and then presents a distributed version of MeeCast. We prove that the distributed solution achieves not only optimality but also fast self-stabilization.

A. Centralized MeeCast

As we have shown in section III, the objective function as well as the constraints are expressed in a linear form of the variables x and d under the ideal TDMA, O-MAC and S-MAC with the non-interfering transmission assumption. The energy efficiency optimization problem is therefore solved via existing linear programming techniques at the base station by collecting global information about A, s and D. The centralized program consists of three phases. First, the base station initiates an information collection process by flooding a start message through the network. This message informs nodes about the start of a new configuration and constructs a routing tree for the information collection in the next phase. In the second phase, each node v_i collects and sends its local connectivity information, D_i, and s_i, to the base station using the tree constructed. In the last phase, the base station solves the linear programming problem and distributes the optimal d^* and x^* to the network.

The program is executed at the beginning of the deployment and has to be re-executed whenever nodes or links fail, the duty cycle constraints change, or the traffic changes. However, when such changes occur, rather than a network wide collection of neighborhood information, a local report from the affected node(s) will be sufficient for the base station to recompute (x^*, d^*) based on a new (A, s, D). As we can see, when changes/faults are frequent, centralized MeeCast entails heavy overhead.

B. Distributed and Self-Stabilizing MeeCast

To enable more robust, efficient, and autonomic adaptation to traffic changes, network changes and faults, we propose a self-stabilizing and distributed MeeCast to problem (P1) under O-MAC. (The distributed protocol for S-MAC is left to future work.) We show that distributed MeeCast achieves optimal energy efficiency via a shortest path tree with a convergence time proportional to the depth of the network, provided the following assumption holds.

$$(\forall\, x \text{ such that } Ax = s, (2\mathcal{I}_i + s'_i)t_s \le D_i\ ), \tag{7}$$

where $(2\mathcal{I}_i + s'_i)t_s$ is the minimum duty cycle requirement for v_i based on (5).

In other words, given an x that satisfies the flow conservation constraint, we assume that the corresponding d can satisfy the duty cycle constraints. Note that this is typically true for existing monitoring applications whose duty cycles are often in the range of 0.1 to 1 percent.

Now, we show the optimality condition under the assumption. Let p be a *path* from v_i to the base station if and only if for $p = (k_1, k_2, k_3 \ldots, k_{m(p)})$, where $m(p)$ is the length of p, $k_1 = i$, $k_{m(p)} = bs$, and $(k_u, k_{u+1}) \in L$, for $u = 1, 2, \ldots, m(p) - 1$. Define $\mathcal{P}_i$ to be the set of possible paths from v_i to the base station. $\mathcal{P}_i$ is determined by A and is thus constant. Let $s_i(p)$, $p \in \mathcal{P}_i$, be the amount of source traffic from v_i sent along path p. For one packet to be sent from v_i to the base station along p, each node in p, except for the first and last node, has to receive and send this packet once, resulting in a total number of $(m_i - 1)$ sends and $(m_i - 1)$ receives, i.e., a total of $2(m_i - 1)$ communications.

Lemma 1. *When $Ax = s$, we have $s_i = \sum_{p\in\mathcal{P}_i} s_i(p)$ and*

$$\sum_{i\in V} \mathcal{I}_i = \sum_{i\in\{V-bs\}} \sum_{p\in\mathcal{P}_i} s_i(p) \cdot (m(p) - 1)\ . \tag{8}$$

Proof: The first equation is obvious, we prove the second equation. Let $\varphi_i(p, j)$ be an indicator function where $\varphi_i(p, j) = 1$ if and only if for some $p \in \mathcal{P}_i$, $v_j \in p \wedge v_j \ne v_i$. One observation is that $\mathcal{I}_i = \sum_{j\in V} \sum_{p\in\mathcal{P}_j} s_j(p)\varphi_j(p, i)$. Then

$$\begin{aligned}\sum_{i\in V} \mathcal{I}_i &= \sum_{i\in V}\sum_{j\in V}\sum_{p\in\mathcal{P}_j} s_j(p)\varphi_j(p, i) \\ &= \sum_{j\in V}\sum_{p\in\mathcal{P}_j}\sum_{i\in V} s_j(p)\varphi_j(p, i) \\ &= \sum_{j\in V}\sum_{p\in\mathcal{P}_j} s_j(p) \cdot (m(p) - 1)\ ,\end{aligned}$$

where the last equality holds because there are only $m(p) - 1$ nodes with $\varphi_j(p, i) = 1$ for any $p \in \mathcal{P}_j$. ■

Define m^*_i to be the shortest path from node i to the base station, i.e.,

$$m^*_i = \min_{j\in V}(w_{ij} + m^*_j)\ , \tag{9}$$

where w_{ij} =1 if $(i, j) \in L$ and $w_{ij} = \infty$ otherwise. For any $p \in \mathcal{P}_i$, $m^*_i \le m(p)$. Now we present the optimality condition under assumption (7):

Theorem 2. *When ideal TDMA is used and assumption (7) holds, $Q_c(x, d)$ is minimized if and only if for any $i \in V$ and $p \in \mathcal{P}_i$:*

$$m(p) = m^*_i \quad \text{and} \quad d_i = (2\mathcal{I}_i + s'_i)t_s\ . \tag{10}$$

Proof: Feasibility for $d_i \le D_i$ and $Ax = s$ is guaranteed by the assumption and the protocol respectively, it remains to prove optimality. Let x^* be constructed according to $m(p) = m^*_i$ and d^* is set according to this theorem. Let $(x, d) \in \{\Omega(A, s, D) - \{(x^*, d^*)\}\}$ be any feasible solution. Based on inequality (5) and Lemma 1, we have

$$Q_c(x, d) = \frac{\sum_{i\in V} d_i}{\sum_{i\in V} s'_i} \ge \frac{\sum_{i\in V}(2\mathcal{I}_i + s'_i)}{\sum_{i\in V} s'_i} \tag{11}$$

$$= 2\frac{\sum_{i\in V}\sum_{p\in\mathcal{P}_i} s_i(p)(m(p)-1)}{\sum_{i\in V} s'_i} + 1$$

$$\geq 2\frac{\sum_{i\in V} s'_i(m^*_i - 1)}{\sum_{i\in V} s'_i} + 1 \ .$$

As we can see, $Q_c(x, d)$ is minimized when the shortest paths are selected so that m and d satisfy the conditions specified in the theorem. ■

Corollary 1. *If x consists of a shortest path tree, it is feasible and optimal.*

In the following subsections, we first introduce some preliminaries of self-stabilization and then present the distributed MeeCast protocol.

1) Computation Model: We use the read/write computation model. Kulkarni et al show in [13] that the Write All with Collision (WAC) model, which captures communication collisions in WSNs, can be transferred to a read/write model while preserving self-stabilization property if the system is timed, i.e., each node has its own clock and the rate of all clocks are the same. Since we design our protocol under ideal TDMA, the pre-condition of the timed system is satisfied.

The protocol executes at each node and consists of a finite set of variables and actions. Variable z of node i is denoted as $z.i$. Each node contains one set of *public variables* that can be read by neighbors and one set of *copy variables* that store the latest updates of the public variables for each of its neighbors. Each *action* consists of two parts: guard and the statement. We associate a unique name with each action. Thus, an action has the following form:

$$\langle name\rangle :: \langle guard\rangle \longrightarrow \langle statement\rangle \ .$$

The *guard* is a boolean expression over public variables, copy variables, and constant variables of the node. The action updates zero or more variables of the node. An action is *enabled* if and only if its guard evaluates to *true*. Execution of statements is *weakly fair*, i.e., the number of steps between two executions of an infinitely enabled statement is bounded. A statement is executed atomically and at most one action can be executed at a node at one time. Nodes can execute enabled statements concurrently. In a *read action*, v_i reads the public variables from a neighbor j and updates its local copies.

The *state* of v_i is the union of all the variables of v_i. The state of system G, denoted as $q.G$, is the union of the current system topology (V, L) and the current states of all the up nodes. Given a system G and a problem specification, there exist a set of *legitimate* system states, denoted as $Q(G)$. ($Q(G)$ for our problem is specified in section IV-B5.) A system *computation* is a sequence, be it finite or infinite, of alternating states and actions starting with an initial state q_0, where for every $k \geq 1$, state transition q_{k-1}, a_k, q_k means that the execution of action a_k changes the system state from q_{k-1} to q_k. A set of states is *closed* if, starting from any state in this set, no computation has any state outside this set.

Definition 1. *A protocol is self-stabilizing if, starting from an arbitrary initial state q_0, every computation has a suffix where every state is in $Q(G)$.*

2) Fault Model: In a wireless sensor network, nodes and links that are up can crash, nodes and links that are down can become up and join the system. The non-constant variables of any node can be corrupted. We assume that the base station's state, all nodes' constant variables, as well as their actions are never corrupted. When a fault occurs, a system might transit into a state outside of $Q(G)$. Self-stabilization ensures that when faults stop occurring, subsequent execution of the protocol will eventually reach a state in $Q(G)$, from which point the specification of the system will be respected (at least until the next fault occurs).

3) Protocol Overview: Distributed MeeCast stabilizes both x and d for all nodes to $Q(G)$ in a finite number of steps. The routing tree is determined by the m_i at each node. The update of metric m is propagated from the base station as each node reads from its neighbors. Each node locally selects its parent based on the shortest path rule. Because each node keeps a copy of the parent information of its neighbors, it can deduce the set of children and compute x based on the $Ax = s$ constraint and d as shown in Theorem 2. The leaf nodes are the first ones to stabilize x. Then non-leaf node can compute its x once all its children have their x values stabilized.

4) The Design of MeeCast: Our description begins with the constants and variables used in the protocol. MeeCast has four public variables: $m.i$, $d.i$, $x.j.i$, and $p.i$. Node i stores the current distance to the base station in $m.i$; $d.i$ stores node i's current duty cycle; $x.j.i$ is the data rate from v_i to v_j, in which $x.i.i = s'_i$; $p.i$ is the parent of node i. Also each node maintains three copy variables $m'.j.i$, $x'.j.i$, and $p'.j.i$ that stores $m.j$, $x.i.j$, and $p.j$ respectively. In particular, $m'.i.i = m.i$, $x'.i.i = s'_i$, and $p'.i.i = p.i$.

MeeCast also uses two constant variables: $N.i$ and $w.j.i$. $N.i$ is the set of the neighboring nodes of i; $w.j.i$ is the cost of link (i, j) where $w.j.i = 1$ if both v_i and v_j are up and link (i, j) is up. Otherwise, $w.j.i = \infty$. We assume that there's an underlying service that will update $w.j.i$ for each node i. The update of $w.j.i$ can be implemented by various mechanisms such as periodic heartbeat messages. To avoid self-loops, $w.i.i$ is set to ∞. Last, we utilize a helper function *sumRate(i)*, $sumRate(i) = x.i.i + \sum_{j\in N.i \wedge p'.j.i=i} x'.j.i$, to compute $\mathcal{I}_i + s'_i$.

For base station bs, $m.sb$, $d.bs$, and $p.bs$ are always fixed to 0, 1, and -1 respectively, where we assume no node has an ID of -1 . The distributed MeeCast is described in guarded command notation as follows:

Program.i, for node $i \neq bs$ and $j \in N.i$

```
S1 ::    $MP.i \longrightarrow$
             $p.i,\ m.i,\ x.(p.i).i := i,\ \infty,\ 0$
S2 ::    $(m'.j.i + w.j.i < m.i)\ \wedge\ p'.j.i \neq j \longrightarrow$
             $p.i,\ m.i := j,\ m'.j.i + w.j.i$
             $x.j.i := sumRate(i)$
SYN ::   $x.(p.i).i \neq sumRate(i) \wedge \neg MP.i \longrightarrow$
             $x.(p.i).i := sumRate(i)$
             $d.i := (2 * sumRate(i) - x.i.i) * t_s$
RD.j ::  $x'.j.i \neq x.i.j \vee m'.j.i \neq m.j \vee p'.j.i \neq p.j \longrightarrow$
             $p'.j.i,\ m'.j.i,\ x'.j.i := p.j,\ m.j,\ x.i.j$
```

where $MP.i$ is defined as:

$$(\forall k : m.i < m'.k.i + w.k.i) \vee (p.i \neq bs \wedge p'.(p.i).i = p.i) .$$

When $MP.i = true$, node i is a *false sink* whose $m.i$ is less than $w.j.i + m.j$ for any of its neighbors v_j. Upon detecting a false sink, S1 signals the fault by setting the parent of the faulty node to itself so that the descendants will switch to other parents. Also, S1 sets the outgoing rate from the faulty node to 0 so that its ancestors can adjust duty cycles to reflect the removal of the subtree rooted at this faulty node. Action S2 updates the routing structure according to the shortest path rule. To adapt to faulty nodes, faulty links and change of routing, each node updates their $\vec{x}.i$ and $d.i$ whenever they are inconsistent with the latest updates received from the neighbors in action SYN. A node reads from a neighbor j and updates its local copy variables $x'.j.i$, $m'.j.i$, and $p'.j.i$ in action RD.j. Faulty nodes and links are detected when $w.j.i$ becomes ∞.

5) Stabilization: While we relegate the detailed proof of self-stabilization and convergence time analysis to [11], conceptually, the stabilization is a convergence "stair" consisting of three phases, with three corresponding closed sets of states, namely $Q1(G)$, $Q2(G)$, $Q(G)$, each of which is a subset of the previous one. They are defined as follows:

$$\begin{aligned} Q1 &= (\forall\ i \in V : MP.i = false \vee p.i = i) , \\ Q2 &= Q1 \wedge (\forall\ i \in V : m.i = m_i^*) , \\ Q &= Q2 \wedge (Ax = s \wedge d = (2\mathcal{I} + s)t_s) . \end{aligned}$$

$Q(G)$ is the legitimate set of states the system is designed self-stabilize to. We prove stabilization by showing that, starting from an arbitrary state, the system will first self-stabilize to a state in $Q1$, from which the system further self-stabilizes to $Q2$ and finally arrives in Q. $Q1$ ensures that there is no false sink. Starting from any state in $Q1$, the system can self-stabilize to the shortest path tree that grows out from the only sink — the base station. Then starting from any state in $Q2$, x will be stabilized from the leaf nodes and duty cycles are set according to the stabilized x, ending in Q. We can show that the time for the stabilization to $Q1$, $Q2$, Q are all upper bounded by some constant factor of network depth $\delta(G)$, where $\delta(G) = \max_{i \in V}\{m_i^*\}$. Therefore, the overall convergence time is also proportional to $\delta(G)$.

V. Performance Evaluation

In this section, we compare centralized MeeCast with Dozer [14], which is designed to minimize energy consumption for ultra low duty cycle data collection in WSNs. We show that, by appropriately setting D_i, an application programmer can optimize energy efficiency with the guarantee that the application lifetime is above some threshold c. First, let us define the application lifetime $T_l(d)$. Let $\mathcal{E}_i$ be the remaining energy of v_i, then $\mathcal{E}_i/d_i$ measures the remaining lifetime of v_i. Given d, define $T_l(d)$ to be the time when the first node runs out of battery, i.e.:

$$T_l(d) = \min_{i \in V}(\mathcal{E}_i/d_i) .$$

If we set $D_i = \mathcal{E}_i/c$ for all nodes, then we are guaranteed that the application lifetime is greater than c. In the following sections, we compare both the energy efficiency and the application lifetime of MeeCast with Dozer under O-MAC and S-MAC assuming $\mathcal{E}_i = cD_i$ for some constant $c > 0$.

A. O-MAC

Dozer tries to minimize energy consumption by orchestrating MAC, topology control and routing layers. Dozer's scheduler provides another approximated implementation of the ideal TDMA. Each node is aware of the transmission schedules of both its parent and its children so that they send and receive only at the designated slots. However, collisions can occur in Dozer and is explicitly addressed by acknowledgments and random schedule shifts. As we have shown, O-MAC approximates ideal TDMA under low traffic. To compare MeeCast and Dozer on a fair ground, we assume that the random shifts can separate all the interfering communications apart and call this Dozer-Ideal. In other words, the scheduler of Dozer also satisfies the all properties in (4). Dozer assigns 1 as the cost for each link and utilizes a shortest path tree.

Dozer-Ideal and MeeCast optimize (x, d) upon the problem defined by A, s and D. d_i's are computed based Theorem 2. Our results do not include infeasible solutions. As we have proven in Corollary 1, the shortest path scheme achieves the upper bound for energy efficiency E_c. However, Dozer-Ideal achieves such energy efficiency at the cost of violating the nodal energy constraints, resulting in poor application lifetime.

Fig.1a and Fig.1b show the relationship between energy efficiency, application lifetime, and the network size. Each node has an average neighborhood size of 9. As we can see, both E_c and T_l decrease as the network size increases. E_c deceases as the network size grows because the average distance from the base station increases. As the network size increases, the application lifetime gap between Dozer-Ideal and MeeCast increases because Dozer-Ideal tends to route traffic through the same set of bottleneck nodes whereas MeeCast tends to balance the traffic according to duty cycle constraints. T_l attained by Dozer-Ideal reaches as low as 50%

of centralized MeeCast when the number of nodes surpasses 100.

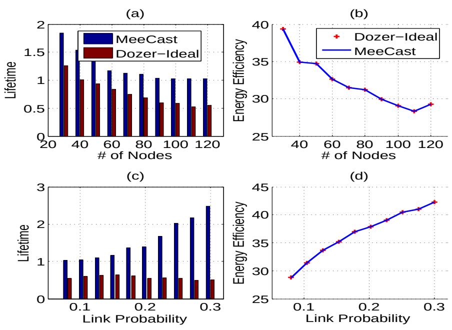

Fig. 1. Energy Efficiency and Lifetime vs. Network Size and Density under O-MAC. Network topologies, i.e. matrix A, are randomly generated. s_i' and D_i are uniformly distributed in $[0, 6]$ packets per minute and $[0.1\%, 1\%]$ respectively.

In Fig.1c and Fig.1d, we study the relationship between energy efficiency, application lifetime, and network density on a network with 100 nodes. Network density is controlled by varying the link probability, i.e. the probability that any two nodes are connected. As we can see, E_c in both Dozer and MeeCast increases as network density increases because the average distance to the base station decreases. T_l in MeeCast increases with link probability because MeeCast can better balance the traffic when each node has more alternative shortest paths. However, T_l in Dozer decreases as the network density increases because bottleneck nodes tend to receive more traffic as the network becomes more connected. T_l of Dozer-Ideal reaches as low as 20% of MeeCast when the average neighborhood size reaches 30 for a link probability of 0.3.

B. S-MAC

In this section, we present simulation results for the same two cases under S-MAC. For comparison, we replace the internal TDMA scheduler of Dozer with S-MAC and call it Dozer-SMAC. Fig.2a and Fig.2b illustrate the relationship between E_c, T_l, and network size. As the number of nodes in the network increases, again both E_c and T_l decrease as the average distance from nodes to the base station increases. Under S-MAC, MeeCast outperforms Dozer-SMAC for not only T_l but also E_c because the duty cycle nodes run at is determined by the neighborhood with the most traffic due the unique shared wakeup period of S-MAC. Since Dozer-SMAC routes traffic according to the shortest paths, nodes close to the base station are likely to take large portions of the network traffic, resulting in neighborhoods with high traffic loads. Thus, the traffic of the neighborhood with most traffic in Dozer-SMAC is likely to be greater than that in MeeCast, resulting in a higher shared duty cycle of all the nodes and lower T_l and E_c.

Fig.2c and Fig.2d illustrate the relationship between E_c, T_l, and network density. As we can see, as the link probability increases, the size of the neighborhood of each node grows and the thus traffic generated per neighborhood increases.

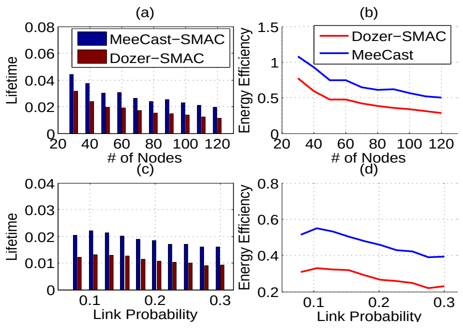

Fig. 2. Energy Efficiency and Lifetime vs. Network Size and Density under S-MAC. s_i' and D_i are uniformly distributed in $[0, 1]$ packets per minute and $[3\%, 7\%]$ respectively.

As a result, the E_c decreases as the required shared duty cycle of the neighborhood increases as each node has to be awake during all communications of its neighbors in the worst case. When network is sparser, the average number of hops from nodes to the base station is higher and the lifetime gap between the two schemes is larger because MeeCast can better balance traffic based on duty cycle upper bounds in a sparser network whereas Dozer-SMAC tends to exacerbates the traffic condition in bottleneck regions.

Overall, even with lower source rates and higher capacity upper bounds, S-MAC achieves lower E_c and T_l than O-MAC in part because of the significant overhearing caused by the shared wakeup scheme.

VI. Related Work

WSN designs aim to meet multiple and sometimes application specific requirements. Nevertheless energy efficiency is typically a critical metric for truly wireless operation. For instance, the design of extant WSN MAC protocols focuses on scheduling so as to minimize idle listening and/or overhearing and delegate the control over duty cycle to upper layers. Protocols such as B-MAC, S-MAC, and O-MAC ([15], [12], [7]) are generic and do not assume any underlying communication pattern such as convergecast. For this reason, they all provide hooks through which higher layers can control the duty cycle depending upon the traffic. However, not all MACs expose such hooks: D-MAC [16] controls duty cycle by itself. Starting from leaf nodes, D-MAC assigns staggered transmissions schedules to nodes in the collection tree. Nodes sends out control packets to update their parents about traffic status and parents adjust their duty cycles accordingly.

Routing too has been extensively studied from the perspective of minimizing the energy consumption of the network. MintRoute [17] actively measures link qualities, updates the routing using a shortest path scheme and adapts to link and node changes. Shah et al [18] show that alternatively switching between optimal and suboptimal paths helps to increase network lifetime while maintaining an energy efficient routing. However, neither of these exploit the underlying MAC protocol or take into account the duty cycle constraints of nodes.

In contrast to optimizing energy efficiency with respect to a single layer, some solutions consider multiple layers. AEM [19] derives traffic information based on tasks provided by application programmer and decides a TDMA-based schedule for all involved nodes at compile time. Work along these lines has often eschewed node energy constraints. It has also integrated multiple layer components (such as time synchronization, routing, and scheduling) into a single monolithic framework, which risks complex designs, and incompatibility with extant MAC protocols.

Mathematical optimization techniques, such as convex optimization, have been proven to be powerful in solving network optimization problems because they enable rigorous cross-stack reasoning of network behaviors that are very difficult otherwise. The network optimization problem is usually formulated as a utility maximization problem with some chosen optimization variables. The network stacks upon which the problem is being solved determine the set of optimization variables among which link rates, routing and scheduling are the most common ones ([5], [20]). Our work falls into the layered network optimization category and addresses the optimization in a context of realistic MAC protocols.

The linear problem we have formulated falls into the class of quasi polymatroidal problems [21] which generalize of the classic minimum cost flow problem by having constraints on nodes instead of links. However, as far as we are aware, the best known centralized algorithm that produces exact solutions achieves a time complexity no better than $O(V^3 \log V)$, which indicates slow convergence time for distributed solutions. In contrast, one consequence of the ultra-low traffic assumption for the distributed version of MeeCast is its fast convergence time.

VII. Conclusion and Future Work

Different network layers are interdependent viz-a-viz energy efficiency, thus network energy efficiency optimization should be addressed in a cross-layer manner while preserving inherent structure to keep complexity manageable. This work has taken a first step in designing a middle layer between standard routing layer and MAC layer designs to optimize energy efficiency for convergecast. It shows that the optimization problem is linear with the assumption that O-MAC and S-MAC achieve interference-free, reliable communication underlow traffic. And its simulations show that our centralized solution MeeCast performs favorably compared with a state-of-the-art convergecast protocol, Dozer.

We have shown that optimality is achieved when the routing structure is a shortest path tree, for cases where the duty cycle constraint can be ignored. Distributed MeeCast suffices for these cases and can work autonomically in the background to self-stabilize the network to the optimal solution in a time proportional to the depth of the network.

Our middle layer can be adapted for other choices of routing protocols and MAC protocols. In future work, we seek to address the class of other protocols for which efficient solutions exist via linear or convex optimization techniques. We will also study non TDMA-style MAC protocols further, for which we would account for interference and investigate the feasibility of distributed solutions.

References

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda *et al.*, “A line in the sand: a wireless sensor network for target detection, classification, and tracking,” *Computer Networks*, vol. 46, no. 5, pp. 605–634, 2004.

[2] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, “An analysis of a large scale habitat monitoring application,” in *Proc. of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 214–226.

[3] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay *et al.*, “A macroscope in the redwoods,” in *Proc. of the 3rd international conference on Embedded networked sensor systems*, 2005, pp. 51–63.

[4] K. Szlavecz, A. Terzis, S. Ozer, R. Musaloiu-E, J. Cogan, S. Small, R. Burns, J. Gray, and A. Szalay, “Life under your feet: An end-to-end soil ecology sensor network, database, web server, and analysis service,” *Arxiv preprint cs.DB/0701170*, 2007.

[5] L. Xiao, M. Johansson, and S. Boyd, “Simultaneous routing and resource allocation via dual decomposition,” *Communications, IEEE Transactions on*, vol. 52, no. 7, pp. 1136–1144, 2004.

[6] M. Chiang, S. Low, A. Calderbank, and J. Doyle, “Layering as optimization decomposition: A mathematical theory of network architectures,” *Proc. of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[7] H. Cao, K. Parker, and A. Arora, “O-MAC: A Receiver Centric Power Management Protocol,” in *ICNP. Proc. of the 14th IEEE International Conference on Network Protocols*, 2006.

[8] J. Zhao and R. Govindan, “Understanding packet delivery performance in dense wireless sensor networks,” in *Proc. of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 1–13.

[9] R. Madan, S. Cui, S. Lall, and N. Goldsmith, “Cross-layer design for lifetime maximization in interference-limited wireless sensor networks,” *IEEE Transactions on Wireless Communications*, vol. 5, no. 11, pp. 3142–3152, 2006.

[10] L. Qian, Y. Zhang, and J. Huang, “MAPEL: Achieving global optimality for a non-convex wireless power control problem,” *Arxiv preprint arXiv:0805.2675*, 2008.

[11] A. A. W. Zeng and N. Shroff, “Maximizing energy efficiency for convergecast via joint routing and duty cycling optimization,” The Ohio State University, ftp://ftp.cse.ohio-state.edu/pub/tech-report/2009/TR36.pdf, Tech. Rep. OSU-CISRC-7/09-TR36, July 2009.

[12] W. Ye, J. Heidemann, and D. Estrin, “Medium access control with coordinated adaptive sleeping for wireless sensor networks,” *Networking, IEEE/ACM Transactions on*, vol. 12, no. 3, pp. 493–506, 2004.

[13] S. Kulkarni and M. Arumugam, “Transformations for write-all-with-collision model,” *Computer Communications*, vol. 29, no. 2, pp. 183–199, 2006.

[14] N. Burri and R. Wattenhofer, “Dozer: ultra-low power data gathering in sensor networks,” in *Proc. of the 6th international conference on Information processing in sensor networks*, 2007, pp. 450–459.

[15] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *Proc. of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 95–107.

[16] G. Lu, B. Krishnamachari, and C. Raghavendra, “An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks,” in *Parallel and Distributed Processing Symposium, 2004. Proc.. 18th International*, 2004.

[17] A. Woo, T. Tong, and D. Culler, “The MintRoute protocol, tinyos-1. x/tos/lib.”

[18] R. Shah and J. Rabaey, “Energy aware routing for low energy ad hoc sensor networks,” in *2002. WCNC2002*, vol. 1, 2002.

[19] O. Gnawali, J. Na, and R. Govindan, “Application-Informed Radio Duty-Cycling in a Re-Taskable Multi-User Sensing System,” in *IPSN 2009, San Fransico, California, USA*, 2009.

[20] J. Kim, X. Lin, N. Shroff, and P. Sinha, “On Maximizing the Lifetime of Delay-Sensitive Wireless Sensor Networks with Anycast,” in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 807–815.

[21] K. C. Ol and M. Kochol, “Quasi polymatroidal flow networks,” 1995.

Appendix

In this section, we prove the self-stabilizing property and analyze the convergence time of distributed MeeCast. Define a node to be *privileged* if one or more of its actions is enabled. First we introduce the concept of rounds to measure the time complexity of the protocol. The *first round* of a computation γ is the shortest prefix γ' of γ in which some each continuously privileged node executes at least one step. Let γ'' be the suffix that follows γ', i.e. $\gamma = \gamma'\gamma''$, the the *second round* of γ is the first round of γ''.

Since we assume weak fairness in the execution of statements, the following statements hold: (i) there exists a upper bound R over the the number of computational steps in a round; (ii) for a statement that is continuously enabled, there exists a upper bound M over the number of rounds that this statement is executed. If either of these statements does not hold, the semantics of fair execution is violated. To show the convergence time of the system is proportional to the depth of the network G, denoted as $\delta(G)$, is equivalent to show that the number of rounds the system takes to converge is proportional to $\delta(G)$.

We repeat the definitions of $Q1$, $Q2$, and Q here:

$$\begin{aligned}
Q1 &= \{\forall\, i \in V : MP.i = false \vee p.i = i\} \\
Q2 &= Q1 \wedge \{\forall\, i \in V : m_i = m_i^*\} \\
Q &= Q2 \wedge \{Ax = s \wedge d = (2\mathcal{I} + s)t_s\}
\end{aligned}$$

where $MP.i$ is defined as:

$$(\forall k : m.i < m'.k.i + w.k.i) \vee (p.i \neq bs \wedge p'.(p.i).i = p.i) \ .$$

Lemma 1. *$Q1$, $Q2$, and Q are all closed.*

Proof: Starting from a state in $Q1$, system G will leave $Q1$ if and only if some node becomes a false sink when one of the following transient faults occurs (i) node i's state becomes corrupted; (ii) i's parent becomes corrupted or fails, or (iii) link (i, j) fails. When one of the above involves v_j and v_i reads from v_j via `RD.j`, the state of v_i transits out of $Q1$. Otherwise, no action in MeeCast will transfer a node from a healthy node into a false sink. Thus $Q1$ is closed. $Q2$ is closed because although both `S1` and `S2` modify m, neither of them are enabled in $Q2$. Thus, $Q2$ is closed. For any state in Q, since $Ax = s$ holds, every node should have integrated the traffic generated by the sub-trees rooted at its children as well as its own source traffic into its outgoing traffic. Thus the following must hold

$$\begin{aligned}
&\forall\, i \in V : \\
&\quad (\ (\forall\, j \in N.i : (p'.j.i = p.j \wedge x'.j.i = x.i.j \wedge m'.j.i = m.j)) \\
&\qquad \wedge\, x.(p.i).i = x.i.i + \sum_{k:p.k=i} x.i.k\ ) \ .
\end{aligned}$$

Thus, no node is privileged in Q and Q is closed. ■

Lemma 2. *Starting from an arbitrary state q_0, system G will self-stabilize to a state in $Q1$ in a number of rounds proportional to $\delta(G)$.*

Proof: Define indicator function $mp(i)$ where $mp(i) = 1$ if $MP.i = true \wedge p.i \neq i$ and 0 otherwise. Define step function

$$mp(q.G) = \sum_{i \in V} mp(i) \quad .$$

$mp(q.G)$ decreases whenever `S1` is executed at some node. As long as $mp(G) > 0$, there must exist some node i in which `S1` is continuously enabled. $MP.i$ becomes false once `S1` is executed at node i unless another transient fault occurs. Once a node i executes its `S1` and changes $MP.i$ to *false* by setting $p.i = i$, its children will execute $RD.i$ and learn about the faulty state in at most M rounds. It takes at most another M rounds before all the children execute `S1` to signal the fault. Thus, all nodes k hops away from the base station will be in $Q1$ after at most $2kM$ rounds after the fault occurs. As a result, starting from an arbitrary state q_0, the system G will self-stabilizes to a state in $Q1$ in at most $2M\delta(G)$ rounds. ■

Lemma 3. *Starting from any state in $Q1$, system G will self-stabilize to a state in $Q2$ in a number of rounds proportional to $\delta(G)$.*

Proof: Thus, for any state in $Q1$, the predicate $(\forall\ i \in V, \exists\ j \in N.i : m.i \geq m.j + w.j.i)$ must hold, otherwise v_i is a false sink. Define a function $m(q.G)$ to be:

$$m(q.G) = \sum_{i \in V} m.i$$

Obviously, $m(q.G) \geq \sum_{i \in V} m_i^*$. Every time action `S2` is executed, $m(G)$ will be decreased. For any system not in $Q2$, there must exist some node v_i whose action `S2` is continuously enabled. This is because, once `S2` is enabled at some node, it will continue to be enabled until `S2` is executed. The only other action `S1` that modifies $m.i$ is never enabled in $Q1$. Thus, $m(q.G)$ is guaranteed to decrease every M rounds. And thus the system will stabilize to $Q2$.

Now we show the convergence time using induction. We will prove that the nodes k hops away from the base station will have $m_i = m_i^*$ in at most $2Mk$ rounds. After the first M rounds, any node i that is one hop away from the base station must have learned that $m'.bs.i = 0$ because $RD.bs$ must be continuously enabled at a node i whose $m'.bs.i \neq 0$. Then among these nodes, for any i whose $p.i \neq bs$, `S2` will be executed in at most another M rounds. Thus, after the first $2M$ rounds, every node i one hop away from the base station must have $m.i = m_i^* = 1$. Now assume that after $2kM$ rounds, all nodes k hops away from the base station has set their $m.i = m_i^* = k$. The same rationale can be used to prove that every v_i $k+1$ hops away from the base station will have $m.i = m_i^* = k+1$ in at most another $2M$ rounds. As a result, starting from any state in $Q1$, system G will self-stabilize to a state in $Q2$ in at most $2M\delta(G)$ rounds.

■

Lemma 4. *Starting from any state in $Q2$, system G will self-stabilize to a state in Q in a number of rounds proportional to $\delta(G)$.*

Proof: We only prove the system will converge to a state in Q', where $Q' = Q2 \wedge \{Ax = s\}$. Since d is set according to x in SYN, the stabilization to Q directly follows that to Q'. Define step function $F(q.G)$=(x(q.G),y(q.G)) where $x(q.G)$ is define as:

$$x(q.G) = \sum_{i \in V} \omega(i)$$

where $\omega(i) = 1$ if $x.(p.i).i \neq$ *sumRate(i)* and $\omega(i) = 0$ otherwise; $y(q.G)$ is defined as

$$y(q.G) = \sum_{i \in V} \sum_{j \in V} (\sigma_j(i) \cdot x.i.i - p_j(i) \cdot x.j.i) \ .$$

where $\sigma_j(i) = 1$ if v_j is on the path from v_i to the base station and $\sigma_j(i) = 0$ otherwise; $p_j(i)$ also is an indicator function where $p_j(i) = 1$ if v_j is the parent of v_i and $p_j(i) = 0$ otherwise. $F(q.G) > F(q'.G)$ if and only if $x(q.G) > x(q'.G)$ or $x(q.G) = x(q'.G) \wedge y(q.G) > y(q'.G)$. $\sum_{j \in V}(\sigma_j(i) \cdot x.i.i$ computes the total traffic that should pass v_i given the routing tree defined by $m.i$. When the system stabilizes, the outgoing traffic from each node should be equal to the sum of traffic passing through it.

$x(q.G)$ is non-decreasing as $x.j.i$ is set to *sumRate(i)* in action SYN. Since the equality $m_i = m_i^*$ holds for any state in $Q2$, the child-parent pair defines a shortest path from every node to the base station. Since every node reads the traffic information from its children, the information about the amount of outgoing traffic generated by v_i propagates down the path when the RD.i action is executed at v_i's parent. In other words, $y(q.G)$ decreases whenever some node i's traffic information is propagated further to an ancestor that has not previously included this information in its outgoing traffic calculation. Eventually, the traffic generated by v_i will be factored into the traffic for every node on $v_i's$ path to the base station. This process occurs concurrently at all the paths and continues until the last path finishes and $F(q.G)$ decrease to $(0,0)$. As a result, the time complexity of this process is determined by the longest path in the network, whose length is $\delta(G)$. Between any pair of child and parent, it takes at most M rounds for the parent to read from the child. It takes at most another M rounds for the parent to update its outgoing rate by executing SYN. As a result, it takes at most $2M\delta(G)$ rounds for the system to self-stabilize from any state in $Q2$ to Q ■

Proposition 1. *Based on Theorem 2, the system is optimized when the system state is in Q in a number of rounds proportional to $\delta(G)$.*