Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 106439, 15 pages
doi:10.1155/2012/106439

Research Article

An Energy-Efficient CKN Algorithm for Duty-Cycled Wireless Sensor Networks

Lei Wang,[1] Zhuxiu Yuan,[1] Lei Shu,[2] Liang Shi,[3] and Zhenquan Qin[1]

[1] *School of Software, Dalian University of Technology, Dalian 116621, China*
[2] *Department Multimedia Engineering, Osaka University, Osaka 565-0871, Japan*
[3] *School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China*

Correspondence should be addressed to Lei Shu, lei.shu@ieee.org

Received 21 December 2011; Revised 14 March 2012; Accepted 15 March 2012

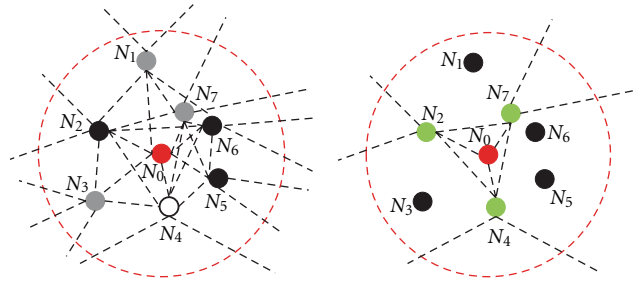
Academic Editor: Yunhao Liu

To prolong the lifetime of a wireless sensor network, one common approach is to dynamically schedule sensors' active/sleep cycles (i.e., duty cycles) using sleep scheduling algorithms. The connected K-neighborhood (CKN) algorithm is an efficient decentralized sleep scheduling algorithm for reducing the number of awake nodes while maintaining both network connectivity and an on-demand routing latency. In this paper, we investigate the unexplored energy consumption of the CKN algorithm by building a probabilistic node sleep model, which computes the probability that a random node goes to sleep. Based on this probabilistic model, we obtain a lower epoch bound that keeps the network more energy efficient with longer lifetime when it runs the CKN algorithm than it does not. Furthermore, we propose a new sleep scheduling algorithm, namely, Energy-consumption-based CKN (ECCKN), to prolong the network lifetime. The algorithm EC-CKN, which takes the nodes' residual energy information as the parameter to decide whether a node to be active or sleep, not only can achieve the k-connected neighborhoods problem, but also can assure the k-awake neighbor nodes have more residual energy than other neighbor nodes in current epoch.

1. Introduction

Wireless sensor networks (WSNs) are normally powered by batteries with limited energy, which are difficult or impossible to be recharged or replaced. A common approach for saving the sensor nodes' energy is to select a subset of nodes to remain active/awake and let others go to sleep in a given epoch. Most of current literatures on sleep scheduling in WSNs are to achieve *point coverage* and/or *node coverage* problems [1]. *Point coverage* problem (also called *spatial coverage*) focuses on selecting a set of active nodes in an epoch so that every point of the deployment space is covered, while considering some optimization goals, for example, minimizing energy consumption [2], minimizing average event detection latency [3]. *Node coverage* problem (also called *network coverage*) focuses on choosing a set of active nodes, in which (1) they construct a connected backbone and (2) sleeping nodes are direct neighbors of at least one active node [4]. This node coverage problem is to ensure that any two nodes in the network can communicate with each other through the connected backbone.

The Connected K-Neighborhood (CKN) algorithm is a distributed sleep scheduling algorithm [1], which can reduce the number of active nodes efficiently. It keeps the network k-connected and optimizes the geographic routing performance. Supporting the geographic routing performance is not studied in any previous *point coverage* and *node coverage* researches. Although, the CKN algorithm performs well with the geographic routing protocols, the following questions are not addressed in paper [1]. (1) *How frequently should the CKN algorithm be executed in the network so that it can really help to save energy, for each time executing the CKN algorithm also consumes energy?* Intuitively, executing the CKN algorithm will consume a mass of energy with substantial data transmission to exchange local information between nodes and their neighbors, which influences the energy consumption distribution of network. (2) *Do all active sensor nodes in the CKN algorithm* [1] *consume the*



(a) Nodes' Residual Energy (b) CKN Active Neighbors

FIGURE 1: Nonuniform energy consumption problem. In (a), the black nodes represent the nodes which have more residual energy than those of node N_0, the grey and white nodes have less energy than N_0's, and the white node has the least energy. In (b), the green nodes represent the active nodes according to the CKN algorithm in a WSN, and the sensor nodes with less residual energy are selected.

energy uniformly in each epoch? We identify the *nonuniform energy consumption* problem, as shown in Figure 1. It is recognized that when executing the CKN algorithm in WSNs, sensor nodes with less residual energy are possible to be selected, which may result in that the energy of these sensor nodes can be fast consumed. The problem is caused by: *the CKN algorithm chooses the active nodes absolutely based on the* ranks *that are randomly given at the beginning of executing the CKN algorithm in each epoch*. In other words, the CKN algorithm cannot ensure the network energy is balancedly consumed.

Motivated by above two major issues, we conduct theoretical studies on two important questions based on the CKN algorithm. The first question is as follows. *Is the CKN algorithm energy saving for any given value of k and the epoch? If not, how frequently should the CKN algorithm be executed so that the network is energy saving?* In order to find out the relationship between the epoch and the energy consumption, we build a probabilistic model for the CKN algorithm to compute the probability that each random node goes to sleep and the expected total number of epochs during each node's lifetime. We formulate the lower bound of an epoch to keep the CKN algorithm energy efficient.

We address the second question based on the analysis for the first problem: *How do we design a new sleep scheduling algorithm based on the CKN algorithm that can balance the energy consumption to prolong network lifetime further?* Satisfying all those requirements that the CKN algorithm holds, a new decentralized sleep scheduling algorithm is challenging. In the light of the discussions for the question 1, we propose a new sleep scheduling algorithm, named energy-consumption-based CKN (EC-CKN), to prolong the network lifetime. The advantage of the EC-CKN algorithm over the original CKN algorithm is that it takes the nodes' residual energy information as parameter to decide whether a node to be active or sleep. The EC-CKN algorithm inherits all the major properties of the CKN algorithm, that is, solving the k-connected neighborhoods problem. Meanwhile, it also makes a significant new contribution to the energy efficiency by assuring the k-active neighbor nodes have more residual energy than other neighbor nodes in the current epoch. A theoretical analysis on the energy consumption of the EC-CKN algorithm is given to show the correctness of the new contribution.

The rest of the paper is as follows. Section 2 shows the network model. Section 3 presents the original CKN algorithm regulation and its properties. Section 4 builds a probabilistic model to compute the probability that a random node goes to sleep. Section 5 presents the EC-CKN algorithm. Section 6 demonstrates the properties of the EC-CKN algorithm. Section 7 shows the simulation results about the original CKN algorithm and the EC-CKN algorithm, comparing theoretical values and simulation results. Finally, Section 9 concludes the paper.

2. Network Model

2.1. Communication Network Model. A multihop sensor network is modeled by a graph $G = (S, E)$, where $S = \{s_1, s_2, \ldots, s_n\}$ is the set of sensor nodes and E is the set of directed links. Each node has a uniform transmission radius of r_t, and the necessary condition of $(s_i, s_j) \in E$ is $|s_i - s_j| \leq r_t$ and a node s_j is the next hop of s_i to the sink by the routing protocol. If $(s_i, s_j) \in E$, we use $l_{i,j}$ to denote (s_i, s_j). Each node also has a uniform interference radius of r_f. An node s_j is interfered by the signal from s_i, if $|s_i - s_j| \leq r_f$ and s_j is not the intended receiver. Let I_i be the interference region that centers at s_i with the interference radius r_f. Each node is only equipped with a single radio interface and has the uniform initial energy $\mathcal{E}_0$. The entire network lifetime is divided into epochs, and each epoch is T. At the beginning of each epoch, a node transmits packets in T_1, and then it runs the sleep scheduling algorithm to decide the state of the next epoch in T_2 (where $T = T_1 + T_2$) as shown in Figure 2.

2.2. Event Generation Model. Assume each node has a uniform sensing radius r_s. Let C_u denote the sensing region of the node s_u, which centers at s_u with the sensing radius r_s. An event occurs when the sensing unit of a node s_u picks up a signal with the power above a predetermined threshold within the sensing region C_u [5]. Suppose the

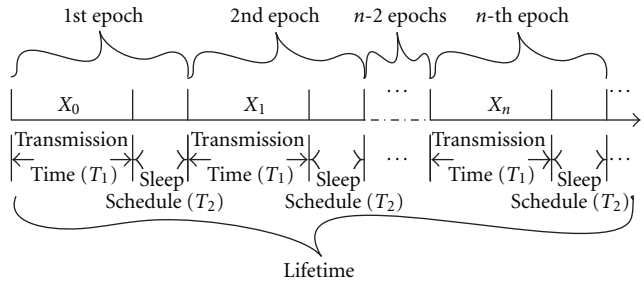


Figure 2: The lifetime of a node s_u consists of many epochs. Each epoch includes packet transmission time and sleep scheduling algorithm's execution time.

temporal event behavior over the entire sensing region, $\mathcal{A}$, is a Poisson process with an average event rate λ. Let $p_{XY}(x, y)$ denote an independent probability distribution of the spatial distribution of events. Let p_e denote the probability that an event is detected by a node s_u, given the fact that it occurred in $\mathcal{A}$:

$$p_e = \frac{\int_{C_u} p_{XY}(x, y)\,dx\,dy}{\int_{\mathcal{A}} p_{XY}(x, y)\,dx\,dy}, \tag{1}$$

where $p_{XY}(x, y)$ is the spatial distribution of events that is characterized by an independent probability distribution. Let $p_m(T, n)$ denote the probability that n events occur in an epoch T at a node s_u. Therefore, the probability of no events occurring in C_u over an epoch T is given by

$$\begin{aligned} p_m(T, 0) &= \sum_{i=0}^{\infty} \frac{e^{-\lambda T(\lambda T)^i}}{i!}(1 - p_e)^i \\ &= e^{-p_e \lambda T}. \end{aligned} \tag{2}$$

Let $p_m(T)$ denote the probability that at least one event occurs in an epoch T at a node s_u:

$$p_m(T) = 1 - p_m(T, 0) = 1 - e^{-p_e \lambda T}. \tag{3}$$

That is the probability of at least one event occurring at the node s_u is an exponential distribution characterized by a spatially weighted event arrival rate $\lambda_{u,u} = \lambda \times p_e$.

2.3. Buffer Analysis. Now, we consider two sources of traffic as an input to the buffer of each node [6].

Generated Packets. The sensing unit of a node senses events and generates packets as discussed in Section 2.2. These packets are *generated packets*. For a node s_u, the rate of the generated packets is denoted by $\lambda_{u,u}$.

Relay Packets. A node also receives packets from its upstream nodes and then forwards them to the sink node. (Along the data stream from a source node to the sink node by the routing protocol, downstream nodes are closer to the sink node, and receive packets sent by the node. Upstream nodes are far away from the sink node, and transmit packets to the node). These packets are referred as *relay packets*. The rate at which a node s_u receives relay packets from a node s_v is denoted as $\lambda_{v,u}$.

Therefore, the input packet rate of s_u's buffer, λ_u, can be written as

$$\lambda_u = \lambda_{u,u} + \lambda_{u,r} = \lambda_{u,u} + \sum_{v \in \mathcal{N}_u^{\text{in}}} \lambda_{v,u}, \tag{4}$$

where $\lambda_{u,r}$ is the total relay packet rate at the node s_u, $\mathcal{N}_u^{\text{in}}$ is the set of nodes that have the node s_u as the next hop, and $\lambda_{v,u}$ is the packet rate from the node s_v to the node s_u. Let γ_u be the output rate of a node, which is given by

$$\gamma_u = (1 + e_u)\lambda_u, \tag{5}$$

where e_i is the packet error rate.

2.4. Channel and Energy Consumption Model. The energy consumption model characterizes energy consumption of a node in the network. Suppose there is no energy consumption when a node is sleep. If a node is active, we classify the energy consumption into three general categories.

(1) The Constant Energy Consumption. is the minimum energy needed to sustain a node when it is active without the packet transmission. It includes, for example, the battery leakage, energy consumed during the state transformation.

(2) The Additional Energy Consumption. is the energy consumed by the data transmission during the sleep scheduling algorithm running time.

(3) The Conventional Energy Consumption. includes the receiving energy consumption and the transmitting energy consumption except the local information exchange in the sleep scheduling algorithm, which is based on the first-order radio model [7].

The energy loss is due to the channel transmission, ϵ_{amp} is the transmit amplifier. And the transmitting energy consumption for a bit packet is

$$E_{tx} = \mathcal{E} + \epsilon_{\text{amp}} \cdot r_t^2, \tag{6}$$

and the receiving energy consumption is

$$E_{rx} = \mathcal{E}, \tag{7}$$

where $\mathcal{E}$ is energy consumed by the transmitter or receiver circuitry.

2.5. Lifetime Definition. There is no universally agreed definition of network lifetime as it depends on the specific application. The lifetime can be measured by the time when the first node exhausts its energy, or when a certain fraction of nodes is dead, or even when all nodes are dead. Alternately, it may be reasonable to measure the network lifetime by application-specific parameters, such as the time when the network can no longer relay sensory data packets. In this paper, we define the network lifetime is the time when the first sensor node run out its energy from the beginning. The general network lifetime is the exact individual lifetime of each active node [8].

Theorem 1. *For a sensor network, each node has nonrechargeable initial energy $\mathcal{E}_0$, the average general network lifetime $\mathbb{E}[\mathcal{L}]$, is given by*

$$\mathbb{E}[\mathcal{L}] = \frac{\mathcal{E}_0}{E_c + \lambda_r \mathbb{E}[E_{rx}] + \gamma \mathbb{E}[E_{tx}]}, \tag{8}$$

where E_c is the constant energy consumption on the first died node, $\mathbb{E}[E_{rx}]$ is the expected receiving energy consumption, and $\mathbb{E}[E_{tx}]$ is the expected transmitting energy consumption.

Proof. Suppose there are M independently and identically distributed trials on the same sensor network to record the network lifetime $\mathcal{L}$, the receiving energy consumption of each bit E_{rx}, and the transmitting energy consumption of each bit E_{tx}. For the mth trial ($1 \le m \le M$), the total energy consumed by the first died node during the whole lifetime is

$$\mathcal{E}_0 = E_c \mathcal{L}^m + \sum_{i=1}^{N_{rx}^m} E_{rx}^m(i) + \sum_{i=1}^{N_{tx}^m} E_{tx}^m(i), \tag{9}$$

where N_{rx}^m is the number of bits to be received, and N_{tx}^m is the number of bits to be transmitted of the first died node during the network lifetime of the mth trial. Summing (9) up over the M trials and dividing both sides by M, we obtain

$$\begin{aligned}\mathcal{E}_0 = \frac{1}{M}\sum_{m=1}^{M} \mathcal{L}^m \Bigg[& E_c \\ & + \left(\frac{\sum_{m=1}^{M} N_{rx}^m}{\sum_{m=1}^{M} \mathcal{L}^m}\right) \times \left(\frac{\sum_{m=1}^{M}\sum_{i=1}^{rx} E_{rx}^m(i)}{\sum_{m=1}^{M} N_{rx}^m}\right) \\ & + \left(\frac{\sum_{m=1}^{M} N_{tx}^m}{\sum_{m=1}^{M} \mathcal{L}^m}\right) \times \left(\frac{\sum_{m=1}^{M}\sum_{i=1}^{N_{tx}^m} E_{tx}^m(i)}{\sum_{m=1}^{M} N_{tx}^m}\right)\Bigg].\end{aligned} \tag{10}$$

Note that $\lim_{M\to\infty}(\sum_{m=1}^{M} N_{rx}^m)/(\sum_{m=1}^{M} \mathcal{L}^m) = \lambda_r$ is the average receiving rate and $\lim_{M\to\infty}(\sum_{m=1}^{M} N_{tx}^m)/(\sum_{m=1}^{M} \mathcal{L}^m) = \gamma$ is the average transmitting rate.

The average receiving energy consumed in the ith received bit can be written as

$$\mathbb{E}[E_{rx}] = \lim_{M\to\infty} \frac{\sum_{m=1}^{M} E_{rx}^m(i)\mathcal{X}_m(i)}{D_{rx}(i)}, \quad 1 \le i \le N_{rx}, \tag{11}$$

where $\mathcal{X}_m(i) = 1$ for $1 \le i \le N_{rx}^m$ and 0 otherwise. $D_{rx}(i) = \sum_{m=1}^{M} \mathcal{X}_m(i)$ is the total number of the occurrence of the ith received bit among the M trials, and $N_{rx} = \max_m\{N_{rx}^m\}$ is the maximum number of received bits during the network lifetime. The probability that the received bit chosen randomly happens to the ith received bit is given by

$$p_{rx}(i) = \lim_{M\to\infty} \frac{D_{rx}(i)}{\sum_{m=1}^{M} N_{rx}^m}, \quad 1 \le i \le N_{rx}. \tag{12}$$

Averaging (11) over the received bit chosen randomly indexing i, the expected receiving energy consumption is defined as

$$\begin{aligned}\mathbb{E}[E_{rx}] &\triangleq \mathbb{E}_{rx}^{i}\{\mathbb{E}[E_{rx}(i)]\} \\ &= \lim_{M\to\infty} \frac{\sum_{m=1}^{M}\sum_{i=1}^{N^m} E_{rx}^m(i)}{\sum_{m=1}^{M} N_{rx}^m},\end{aligned} \tag{13}$$

where $\mathbb{E}[E_{rx}(i)]$ is the average energy consumed in ith bit packet, $\mathbb{E}_{rx}^{i}\{\cdot\}$ denotes the expectation over the randomly chosen received packet indexing i.

Similarly, the expected transmitting energy consumption is

$$\begin{aligned}\mathbb{E}[E_{tx}] &\triangleq \mathbb{E}_{tx}^{i}\{\mathbb{E}[E_{tx}(i)]\} \\ &= \lim_{M\to\infty} \frac{\sum_{m=1}^{M}\sum_{i=1}^{N_{tx}^m} E_{tx}^m(i)}{\sum_{m=1}^{M} N_{tx}^m},\end{aligned} \tag{14}$$

where $\mathbb{E}[E_{tx}(i)]$ is the average energy consumed in i th transmitted packet, $\mathbb{E}_{tx}^{(i)}\{\cdot\}$ denotes the expectation over the randomly chosen transmitted packet indexing i. □

3. A Brief Description of CKN

In [1], the studied WSN is represented as an undirected communication graph $G = (S, E)$. N_u is the set of s_u's neighbors. The connected K-neighborhood problem is defined as (i) each node has at least $\min\{k, |N_u|\}$ active neighbors, which can be called awake neighbors; (ii) all active nodes are connected. To solve the problem, the authors developed a sleep scheduling algorithm: connected K-neighborhood (CKN).

In CKN, each node s_u picks a random rank rank_u, broadcasts the rank_u, and collects its neighbors' ranks in R_u. And then, s_u broadcasts R_u and collects R_v from its neighbors, where $s_v \in N_u$. If s_u or its neighbors has less than k neighbors, s_u will remain awake. Otherwise, s_u computes a subset C_u of N_u that is a set of nodes having rank $<$ rank_u. "Before the node s_u goes to sleep it needs to make sure that all nodes in C_u are connected by nodes with rank $<$ rank_u and each of its neighbors has at least k neighbors from C_u" [1].

The CKN algorithm has the following properties: first, each node s_u (awake or not) with $|N_u|$ neighbors must have at least $\min\{k, |N_u|\}$ awake neighbors in each epoch; second, there is the minimal average number of awake nodes per epoch; finally, awake nodes change from epoch to epoch. For any $k \ge 1$, there are more than $4(k + \ln N)$ neighbors for each node, where N is the number of nodes in the network. Then,

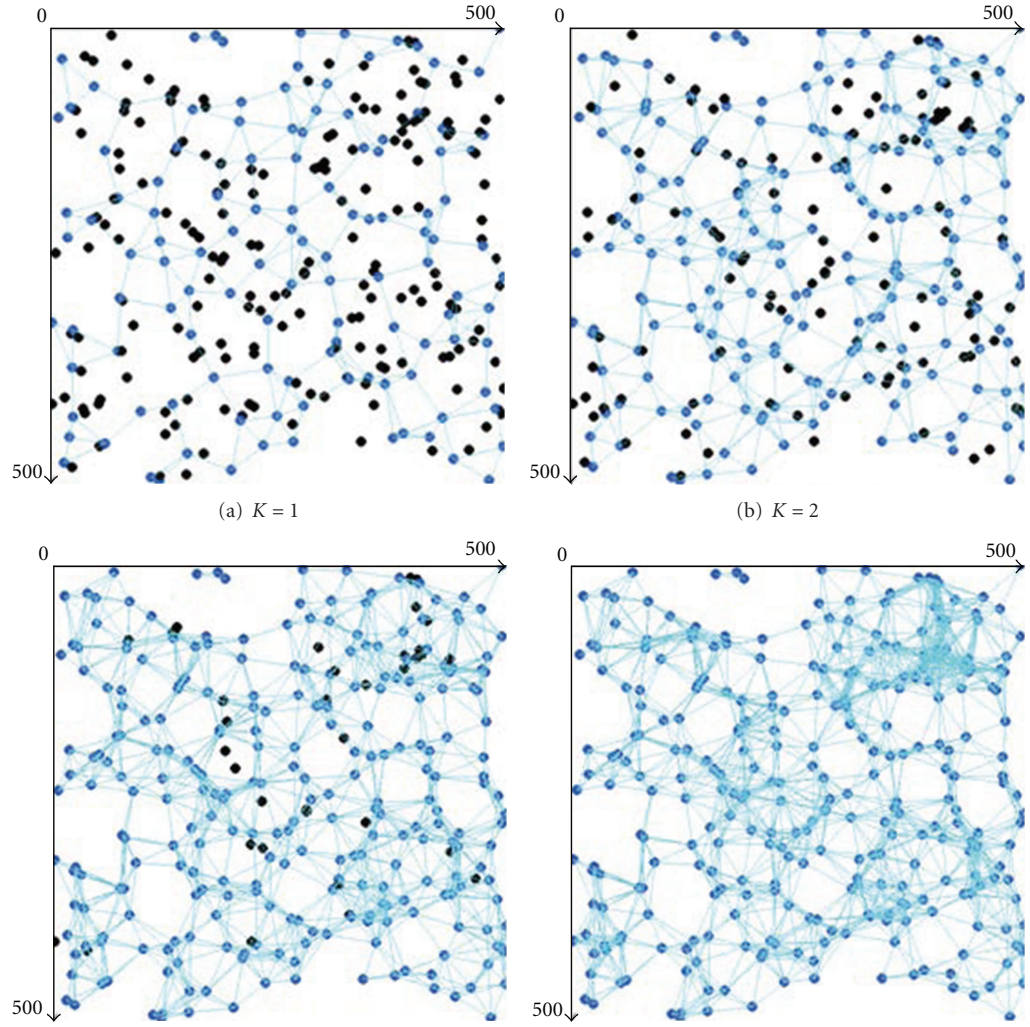


Figure 3: Four examples of executing the CKN algorithm with different values for k. When $k = 1$, a large number of sensor nodes can turn to sleep mode, but, when $k = 8$, almost all sensor nodes have to be always-on. Here, black and unconnected sensor nodes are sleeping nodes.

with high probability, $|\mathrm{CKN}_k| = O(\ln N) \cdot |\mathrm{OPT}_k|$, where $|\mathrm{CKN}_k|$ is the number of awake nodes by the CKN algorithm and $|\mathrm{OPT}_k|$ is the number of awake nodes by an optimal algorithm that finds a minimum connected k-neighborhood as Figure 3.

4. Analysis of the CKN Algorithm

In this section, we build up a probabilistic model for the CKN algorithm to compute the probability that one node s_u goes to sleep in each epoch. Based on the probability, we formulate the lower bound of an epoch to keep the CKN algorithm energy efficient.

Notations used in this section. N_u is the set of s_u's neighbors, and N'_u is the set of s_u's 2-hop neighbors. C_u and C'_u are the subsets of N_u and N'_u having rank $\leq \mathrm{rank}_u$. $|N_u|$, $|N'_u|$, $|C_u|$, and $|C'_u|$ are the number of the elements in N_u, N'_u, C_u, and C'_u, respectively. Graphs G_{C_u} and $G_{C'_u}$ are composed of nodes and potential links in C_u and C'_u.

For a homogeneous Poisson point process in two dimensions, the probability that a random node has n neighbors is [9]

$$P(|N_u| = n) = \frac{(\rho\pi {r_t}^2)^n}{n!} \cdot e^{-\rho\pi {r_t}^2}, \tag{15}$$

where $\rho = N/\mathcal{A}$ is the nodes density. And a node is isolated with a probability of $P(|N_u| = 0) = e^{-\rho\pi {r_t}^2}$. The expectation of the number of s_u's neighbors is

$$\mathbb{E}[|N_u|] = \rho\pi {r_t}^2. \tag{16}$$

If there are at least k different paths connecting any two different vertices in the graph G, the graph is *k-connected*

($k = 1, 2, \ldots$). The probability that the graph G is k-connected is

$$P\left(G^k\right) = \left(1 - \sum_{n}^{k-1} \frac{\left(\rho\pi {r_t}^2\right)^n}{n!} \cdot e^{-\rho\pi {r_t}^2}\right)^N. \tag{17}$$

4.1. Sleep Probability in the CKN Algorithm. For the CKN algorithm, when a node s_u has at least k neighbors, whether it goes to sleep is decided by two factors: (1) "any two nodes in C_u are connected either directly or indirectly through s_u's 2-hop neighbors that have rank less than rank_u" and (2) "any node in N_u has at least k neighbors for C_u" [1].

Lemma 2. *$|C_u| \sim B(|N_u|, p)$ and $|C'_u| \sim B(|N'_u|, p)$ are binomial distributions, where p is the probability that a node has* $\text{rank} < \text{rank}_u$.

Proof. Let $\{\text{rank}_1, \text{rank}_2, \ldots, \text{rank}_{|N_u|}\}$ be the random ranks for nodes in N_u. Suppose $\text{rank}_i \in (0, 1)$ and $i \in [1, |N_u|]$. Let

$$\begin{aligned} x_i &= \begin{cases} 1, & \text{if } \text{rank}_u - \text{rank}_i > 0, \\ 0, & \text{otherwise}, \end{cases} \\ p &= p_i(x_i = 1) = E\left(P_{\text{rank}_u - \text{rank}_i > 0}\right) = \frac{1}{2}, \\ q &= p_i(x_i = 0) = 1 - p. \end{aligned} \tag{18}$$

Let $Z = \sum_{i \in N_u} x_i$, and the probability $z_j = j$ is

$$\begin{aligned} p_j &= \binom{|N_u|}{j} \cdot p^j \cdot q^{|N_u| - j} \\ &= \binom{|N_u|}{j} \cdot p^{|N_u|}, \end{aligned} \tag{19}$$

where $j \in [0, |N_u|]$ and $|C_u| \sim B(|N_u|, p)$. Similarly, $|C'_u| \sim B(|N'_u|, p)$. □

Theorem 3. *Under the CKN algorithm, the sleep probability of a node s_u is*

$$p_s(|C_u|) = \text{Prob}_1 \cdot \text{Prob}_2, \tag{20}$$

and the awake probability is

$$p_a(|C_u|) = 1 - p_s(|C_u|), \tag{21}$$

where Prob_1, the probability a node s_u satisfies the first condition, is defined as

$$\text{Prob}_1 \triangleq \left(1 - e^{-\rho_1 \pi {r_t}^2}\right)^{|C'_u|}, \tag{22}$$

and Prob_2, the probability s_u satisfies the second condition, is defined as

$$\text{Prob}_2 \triangleq P(|C_u| \geq k + 1) \cdot P\left(G^k_{C_u}\right) \cdot P\left(C^{\,k}_{u\,N_u - C_u}\right). \tag{23}$$

Proof. If s_u and its neighbors have at least k neighbors, the two conditions deciding s_u whether to sleep or not can be interpret as the following corresponding conditions: (1) the graph $G_{C'_u}$ is connected, and (2) the graph G_{C_u} is k-connected and each node in the set $N_u - C_u$ has at least k neighbors in C_u.

The probability that the graph $G_{C'_u}$ is k-connected is

$$\begin{aligned} \text{Prob}_1 &\leq \text{Prob}\left(G_{C'_u} \text{ is connected}\right) \\ &\leq P(|N_u|_{\min} \geq 1) \\ &= \left(1 - e^{-\rho_1 \pi {r_t}^2}\right)^{|C'_u|}, \end{aligned} \tag{24}$$

where $|N_u|_{\min}$ is the minimum degree in the graph $G_{C'_u}$, and $\rho_1 = |C'_u|/(4\pi {t_r}^2)$ is the node density in the graph $G_{C'_u}$.

The probability of the condition (2) is

$$\text{Prob}_2 = P(|C_u| \geq k + 1) \cdot P\left(G^k_{C_u}\right) \cdot P\left(C^{\,k}_{u\,N_u - C_u}\right), \tag{25}$$

where $P(G^k_{C_u})$, the probability that the graph G_{C_u} is k-connected, can be expressed as

$$P\left(G^k_{C_u}\right) \leq \left(1 - \sum_{n=0}^{k-1} \frac{\left(\rho_2 \pi {r_t}^2\right)^n}{n!} \cdot e^{-\rho_2 \pi r_t^2}\right)^{|C_u|}, \tag{26}$$

and $P(C^{\,k}_{u\,N_u - C_u})$ is the probability that a node $s_v \in (N_u - C_u)$ has at least k neighbors in C_u, which is

$$P\left(C^{\,k}_{u\,N_u - C_u}\right) = \left(1 - \sum_{n=0}^{k-1} \frac{\left(\rho_2 \pi {r_t}^2\right)^n}{n!} \cdot e^{-\rho_3 \pi {r_t}^2}\right)^{|N_u - C_u|}, \tag{27}$$

where $\rho_2 = |C_u|/(\pi {t_r}^2)$, $\rho_3 = (|C_u| + 1)/(\pi {t_r}^2)$, and $|N_u - C_u|$ is the number of the elements in the set $N_u - C_u$. □

4.2. Energy Consumption of the CKN Algorithm. Based on the result of the probability a random node goes to sleep, we can now analyze the node energy consumption for two cases: (1) it runs the CKN algorithm; (2) it does not run the CKN algorithm.

Lemma 4. *When a node s_u executes the CKN algorithm, its energy consumption is*

$$\mathcal{E}_{\text{ckn}}(s_u) = \mathcal{E}_c^1 + 2(E_{tx} + |N_u| \cdot E_{rx}), \tag{28}$$

and the energy consumption of a node during each epoch is

$$\mathcal{E}_{\text{epoch}}(s_u) = \mathcal{E}_c^2 + T_1(\lambda_r E_{rx} + \gamma E_{tx}), \tag{29}$$

where $\mathcal{E}_c^1$ and $\mathcal{E}_c^2$ are the constant energy consumptions of a node during the time of the CKN algorithm executed and an epoch, respectively.

Theorem 5. *Under the CKN algorithm, the lower bound of an epoch keeping the network energy efficient is*

$$\underline{T} \geq \frac{\mathbb{E}[\mathcal{L}]\left(\mathbb{E}[\mathcal{E}_{\text{ckn}}] + p_a \cdot \mathbb{E}\left[\mathcal{E}_{\text{epoch}}\right]\right)}{\mathcal{E}_0}, \tag{30}$$

where $\mathbb{E}[\mathcal{E}_{\text{ckn}}]$ and $\mathbb{E}[\mathcal{E}_{\text{epoch}}]$ are the expectation energy consumption during the time of the CKN algorithm executed and the expectation of the energy consumption during an epoch.

Input: The least number of neighbors k

Output: Connected k-Neighborhood Network

(1) Get the current residual energy $Erank_u$;

(2) Broadcast $E\text{rank}_u$ and receive the residual energy of its neighbors N_u. Let R_u be the set of the residual energy of nodes in N_u.

(3) Broadcast R_u and receive R_v from each node s_v where $s_v \in N_u$.

(4) If $|N_u| < k$ or $|N_v| < k$ for any $s_v \in N_v$, remain awake.

Return.

(5) Compute $E_u = \{s_v \mid s_v \in N_u \text{ and } E\text{rank}_v > E\text{rank}_u\}$;

(6) Go to sleep if both the following conditions hold. Remain awake otherwise.

(i) Any two nodes in E_u are connected either directly or indirectly through nodes that are the s_u's 2-hop neighbors that have $E\text{rank}_v$ larger than $E\text{rank}_u$;

(ii) Any node in N_u has at least k neighbors from E_u.

(7) Return.

Algorithm 1: Energy-consumption-based CKN (*run the following at each node s_u^*).

Proof. Suppose there are M *i.i.d.* trials on the same network that runs the CKN algorithm as the sleep schedule to record the network lifetime, $\mathcal{L}_{\text{ckn}}$, and the epoch is T in each trial. For the mth trial, the total energy consumed by the first died node during the lifetime is

$$\mathcal{E}_0 = \mathcal{N}_{\text{epoch}}^m \mathcal{E}_{\text{ckn}} + \sum_{i=1}^{\mathcal{N}_{\text{epoch}}^m} \mathcal{E}_{\text{epoch}}^m(i), \tag{31}$$

where $\mathcal{N}_{\text{epoch}}^m$ is the number of epochs in the mth trial. Summing (31) up to the M trials and dividing both sides by M, we obtain

$$\mathcal{E}_0 = \frac{1}{M}\left[\mathcal{E}_{\text{ckn}} \sum_{m=1}^{M} \mathcal{N}_{\text{epoch}}^m + \sum_{m=1}^{M} \sum_{i=1}^{\mathcal{N}_{\text{epoch}}^m} \mathcal{E}_{\text{epoch}}^m(i)\right]. \tag{32}$$

Note that $T\lim_{M\to\infty}(1/M)\sum_{m=1}^{M}\mathcal{N}_{\text{epoch}}^m = \mathbb{E}[\mathcal{L}_{\text{ckn}}]$.

The average energy consumed in the ith epoch can be written as

$$\mathbb{E}\left[\mathcal{E}_{\text{epoch}}\right] = \lim_{M\to\infty} \frac{\sum_{m=1}^{M} E_{\text{epoch}}^m(i)\mathcal{Y}_m(i)}{D_{\text{epoch}}(i)}, \quad 1 \le i \le \mathcal{N}_{\text{epoch}}, \tag{33}$$

where $\mathcal{Y}_m(i) = 1$ if $1 \le i \le \mathcal{N}_{\text{epoch}}^m$ and the node is awake in the ith epoch, and 0 otherwise. $D_{\text{epoch}}(i) = \sum_{m=1}^{M}\mathcal{Y}_m(i)$ is the total number of the ith epoch that the node is awake among the M trials, and $\mathcal{N}_{\text{epoch}} = \max_m\{\mathcal{N}_{\text{epoch}}^m\}$ is the maximum number of epochs during the network lifetime. The probability that a randomly chosen awake epoch of a node happens to the ith epoch is given by

$$p_a(i) = \lim_{M\to\infty} \frac{D_{\text{epoch}}(i)}{\sum_{m=1}^{M}\mathcal{N}_{\text{epoch}}^m} = \lim_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M} p_a(m), \tag{34}$$

where $p_a(m)$ is the nodes awake probability in the mth trial. Averaging (33) over the randomly chosen epoch indexing i, the expected ith epoch energy consumption except for the energy consumed by the CKN algorithm is defined as

$$\mathbb{E}\left[\mathcal{E}_{\text{epoch}}\right] \triangleq \lim_{M\to\infty}\sum_{m=1}^{M}\frac{E_{\text{epoch}}^m(i)\mathcal{Y}_m(i)}{\mathcal{N}_{\text{epoch}}^m}. \tag{35}$$

□

5. The Energy-Consumption-Based CKN Algorithm

We develop a new sleep scheduling algorithm to extend the network lifetime, which can still have all properties of the CKN algorithm.

A scalable distributed solution to the connected k-neighborhoods problem based on the nodes' current residual energy information is challenging for several reasons. First, a node can go to sleep assuming that there are at least k neighbors being awake to keep it k-connected. Second, the outcome of the algorithm must change over epochs so that all nodes have opportunities to sleep. Third, even though nodes decide to sleep or wake up based on their local information, the whole network must be globally connected. The aforementioned three challenges have been achieved by the CKN algorithm [1], which keep the network duty-cycled and connected k-neighborhood. Fourth, awake neighbors of any node s_u have k-top residual. The last one makes sure the energy of the network consumed balancedly, which is the main strength that the EC-CKN algorithm has over the original CKN algorithm.

We address the challenges by proposing the EC-CKN algorithm. The pseudocode of Algorithm 1 depicts the EC-CKN algorithm, which is repeated in each epoch on each node. The algorithm takes an input parameter, k, the required minimum number of per node's awake neighbors. In EC-CKN algorithm, a node s_u broadcasts its current residual energy information $E\text{rank}_u$ (Step 1). It computes a

subset E_u of neighbors having $E\text{rank} < E\text{rank}_u$ (Step 5). Before s_u goes to sleep, it makes sure that any two nodes in E_u are connected either directly or indirectly through the node that is in the s_u's 2-hop neighbors having $E\text{rank} < E\text{rank}_u$, and its neighbors have at least k neighbors from E_u (Step 6). These requirements ensure that when a node has less than k neighbors, none of its neighbors goes to sleep, and when it has more than k neighbors, at least k neighbors decide to remain awake. Note that these requirements are easy to keep by computing locally with 2-hop neighborhood information. The current residual energy is exchanged in Steps 2 and 3.

6. Properties of the EC-CKN Algorithm

This section analyzes the network lifetime, the awake probability, and the energy consumption of the network under the EC-CKN algorithm.

Theorem 6. *For any $k \geq 1$, the average network lifetime of the EC-CKN algorithm increases with the increases of the ratio of the network size N and k, N/k.*

Proof. Suppose N nodes are placed uniformly at random within a deployment area such that the average number of neighbors per node is $\xi \geq 4(k + \ln N)$. Let $\delta = (ck \ln N)/\xi$, for constant $c > 96$ determined by the analysis. Consider executing the algorithm EC-CKN on the network, and let $E\text{rank}^{(i)*}$ be the residual energy of the δth largest residual energy selected by a node in the ith epoch. We claim that, *w.h.p.*, all nodes with residual energy $< E\text{rank}^{(i)*}$ go to sleep. Because there are at most δ nodes with residual energy at least $E\text{rank}^{(i)*}$, we have the average number of the awake nodes in each epoch is

$$|\text{EC}|_k \leq \delta = \frac{ck \ln N}{\xi}. \tag{36}$$

And the average network lifetime under the algorithm EC-CKN can be written as

$$\mathbb{E}[\mathcal{L}_{\text{EC}}] = \frac{NT\mathcal{E}_0}{|\text{EC}|_k \cdot \mathcal{E}_{\text{epoch}} + (N - |\text{EC}|_k) \cdot \mathcal{E}_{\text{EC}}}, \tag{37}$$

where $\mathcal{E}_{\text{EC}} = \mathcal{E}_{\text{ckn}}$ is the energy consumed by executing the EC-CKN algorithm, and T is the length of an epoch. In comparison with the energy consumed by the awake nodes in an epoch, the energy consumed by the sleep nodes in an epoch is considered negligible:

$$\mathbb{E}[\mathcal{L}_{\text{EC}}] \approx \frac{NT\xi\mathcal{E}_0}{ck \ln N \cdot \mathcal{E}_{\text{epoch}}}. \tag{38}$$

□

For the algorithm EC-CKN, a node could have four states: *Init*, *Awake*, *Sleep*, and *Dead*. Let $\mathcal{S} = \{Init = 0, Awake = 1, Sleep = 2, Dead = 3\}$ be the set of the node's states, and $N_s = |\mathcal{S}|$ is the capacity of the states. Nodes can turn into the states *Awake*, and *Sleep* from the states *Init*, *Awake* and *Sleep*, respectively. And the state *Dead* can be only transformed from the states *Awake* and *Sleep*. Figure 4 shows the states transition graph in the algorithm EC-CKN, in which vertices are the states of nodes and the weights of edges are the transition probability between the two states in the ith epoch.

Theorem 7. *Under the algorithm EC-CKN, the difference of the energy consumption between nodes s_u and s_v in the ith epoch is*

$$d^n = \begin{cases} T_2(\Lambda E_{rx} + \Gamma E_{tx}), & \text{if } n = 0, \\ T_2 p_u^a(i-1)(\Lambda E_{rx} + \Gamma E_{tx}), & \text{otherwise,} \end{cases} \tag{39}$$

where Λ and Γ are both the Skellam distributions and $p_u^a(i-1)$ is the average probability that a node is awake in the $(i-1)$-th epoch.

Proof. Let $\mathcal{S}_u = \{\mathcal{S}_u^0, \mathcal{S}_u^1, \ldots, \mathcal{S}_u^i, \ldots\}$ denote the states of s_u in epochs, and let the chain

$$d : d^0, d^1, \ldots, d^i, \ldots \tag{40}$$

denote the difference between the energy consumed by nodes s_u and s_v in each epoch, where $s_v \in N_u$. $d^i = \alpha\mathcal{E}_{v,\text{epoch}}(i) - \beta\mathcal{E}_{u,\text{epoch}}(i)$, where $\mathcal{E}_{u,\text{epoch}}(i)$ and $\mathcal{E}_{v,\text{epoch}}(i)$ are the conventional energy consumption of nodes s_u and s_v if they are awake in the ith epoch. The factors α and β depend on the state of nodes s_u and s_v. If $\mathcal{S}_u^i = 0$ *or* 1, $\alpha = 1$, and if $\mathcal{S}_u^i = 3$, $\alpha = 0$, which is the same with β.

Suppose there are M independently and identically distributed trials. For the mth trial $(1 \leq m \leq M)$, the difference between the energy consumption of the two nodes in the ith epoch is

$$d_m^i = \alpha\mathcal{E}_{v,\text{epoch}}^m(i) - \beta\mathcal{E}_{u,\text{epoch}}^m(i). \tag{41}$$

The average difference of energy consumption in the ith epoch can be defined as

$$d^{(i)} = \lim_{M\to\infty} \frac{T_2}{M}\left[\frac{\sum_{m=1}^{M}\left[\left(\alpha N_{v,rx}^{m,i} - \beta N_{u,rx}^{m,i}\right)E_{rx}\right]}{T_2} + \frac{\sum_{m=1}^{M}\left[\left(\alpha N_{v,tx}^{m,i} - \beta N_{u,tx}^{m,i}\right)E_{tx}\right]}{T_2}\right], \tag{42}$$

where $N_{u,rx}^{m,i}$ and $N_{u,tx}^{m,i}$ are the number of received bits and transmitted bits of the node s_u in the ith epoch during the mth trial. T_2 is the time of the conventional data transmission time of each epoch. We discuss the difference between the energy consumption of two nodes in the following two cases.

Case 1 $(i = 0)$. According to (33) and (14), we obtain

$$d_{(0)} = T_2[(\lambda_{v,r} - \lambda_{u,r})E_{rx} + (\gamma_v - \gamma_u)E_{tx}]. \tag{43}$$

Case 2 $(i > 0)$. According to (33) and (34), we obtain

$$d_{(i)} = T_2 p_u^a(i-1)[(\lambda_{v,r} - \lambda_{u,r})E_{rx} + (\gamma_v - \gamma_u)E_{tx}], \tag{44}$$

where $p_u^a(i-1)$ is the expected probability that the node is awake in the $(i-1)$-th epoch.

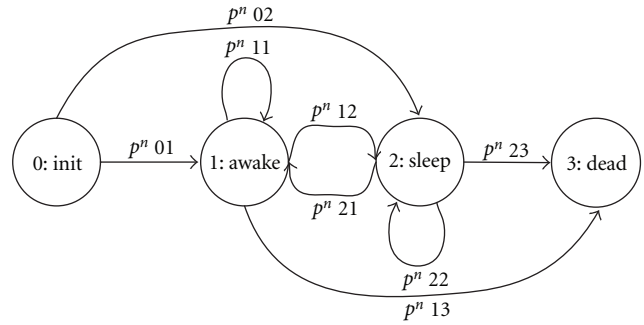


Figure 4: The Markov state transition probability graph of the EC-CKN algorithm.

Note for $\lambda_{v,r}$ and $\lambda_{u,r}$ are two independent Poisson distributions, $\Lambda = \lambda_{v,r} - \lambda_{u,r}$ is the Skellam distribution. Similarly, $\Gamma = \gamma_v - \gamma_v$ is also the Skellam distribution. □

Theorem 8. *Consider a random node s_u, which has more than k neighbors, the probability that a node s_u is awake in the ith epoch under the EC-CKN algorithm is*

$$p_a^i = \begin{cases} p_a(|E_u^0|), & \text{if } i = 0, \\ p_a(|E_u^i|), & \text{otherwise.} \end{cases} \tag{45}$$

Proof. We introduce a new chain

$$D : 0, d^0, D^0, d^1, D^1, \ldots, d^i, D^i, \ldots, \tag{46}$$

to denote the difference between the residual energy of the two nodes s_u and s_v, where $s_v \in s_u$. Then, we obtain

$$D^i = \begin{cases} d^0, & \text{if } i = 0, \\ D^{(i-1)} + d^0, & \text{otherwise.} \end{cases} \tag{47}$$

Therefore, the number of elements in the set E_u^i is $|E_u^i| = |N_u|\varrho^i$, where

$$\varrho^{(i)} = \begin{cases} \dfrac{D^0}{\max\{d^0\} - \min\{d^0\}} & \text{if } i = 0, \\ \dfrac{D^i}{\max\{D^i\} - \min\{D^i\}} & \text{otherwise.} \end{cases} \tag{48}$$

□

Theorem 9. *Under the EC-CKN algorithm, the upper bound of the network lifetime is*

$$\overline{\mathcal{L}}_{\text{EC-CKN}} = \frac{\mathcal{E}_0}{\min \sum_i \sum_a \sigma_{i,a} \mathcal{E}_{\text{epoch}}^n}, \tag{49}$$

where $\sigma_{i,a}$ is the steady-state probability that the action a is chosen when the chain is in the state i under the policy $\mathcal{K}$.

Proof. Now, we construct a Markov State Decision chain for each node s_u:

$$\mathcal{S}_u^0, a^0, \mathcal{S}_u^1, a^1, \ldots, \mathcal{S}_u^n, a^n, \ldots, \tag{50}$$

where $\mathcal{S}_u^0 = 0$, and $\mathcal{S}_u^n \in \{1, 2, 3\}$ denotes the state in the nth epoch ($n \geq 1$), and a^n is the action under the state $\mathcal{S}_u^n$

$$a^n = \begin{cases} \mathcal{E}_{u,\text{epoch}}^n, & \text{if } n = 0 \text{ or } \mathcal{S}_u^n = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{51}$$

Let the policy $\mathcal{K} = \{\kappa_{u,i}^n(a)\}$ denote the probability that the action a is chosen when $\mathcal{S}_u = i$, which satisfies the following two conditions:

$$\begin{aligned} 0 \leq \kappa_{u,i}(a) \leq 1, \quad & \forall i, a, \\ \sum_a \kappa_{u,i}(a) = 1, \quad & \forall i. \end{aligned} \tag{52}$$

Under the policy $\mathcal{K}$, the sequence of states $\mathcal{S}_u^n$ constitutes a Markov chain with the transition probability $p_u^n(\mathcal{K}, i, j)$, which can be written as

$$\begin{aligned} p_u^n(\mathcal{K}, i, j) &= p_u\{\mathcal{S}_u^{n+1} = j \mid \mathcal{S}_u^n = i\} \\ &= \sum_a p_u^n(i, j)\kappa_{u,i}(a). \end{aligned} \tag{53}$$

For the policy $\mathcal{K}$, let $\sigma_{i,a}$ denote the steady-state probability that the chain is in the state i and the action a is chosen:

$$\sigma_{i,a} = \lim_{n \to \infty} P_{\mathcal{K}}\{\mathcal{S}_u^n = i, a^n = a\} \tag{54}$$

The vector $\sigma = \{\sigma_{i,a}\}$ satisfies

$$\begin{aligned} &\text{(i) } \sigma_{i,a} \geq 0, \quad \forall i, a, \\ &\text{(ii) } \sum_i \sum_a \sigma_{i,a} = 1, \quad \forall i, a, \\ &\text{(iii) } \sum_a \sigma_{j,a} = \sum_i \sum_a \sigma_{j,a} p_{i,j}, \quad \forall i, a, j, \end{aligned} \tag{55}$$

Equations (55)(i) and (55)(ii) are obvious, and (55)(iii) follows as the left-hand side equals the steady-state probability of being in the state j and the right-hand side is the same probability computed by conditioning on the state and action chosen one epoch earlier.

Suppose that a reward $\mathcal{R}(Y_i^n, a_i^n) = a^n$ is earned whenever the action a_i^n is chosen in the state i in the n'th epoch. Since $\mathcal{R}(Y_i^n, a_i^n)$ denotes the reward earned at the

epoch n, the expected average reward per epoch under the $\mathcal{K}$ policy can be written as

$$\begin{aligned}\mathbb{E}[\mathcal{R}(\mathcal{K})] &= \lim_{n\to\infty} \frac{\sum_n \sum_{i\in\mathcal{S}} \mathcal{R}(Y_i^n, a_i^n)}{n} \\ &= \sum_i \sum_a \sigma_{i,a_i^n} \mathcal{R}(Y_i^n, a_i^n) \\ &= \sum_i \sum_a \sigma_{i,a_i} a^n.\end{aligned} \tag{56}$$

Therefore, $\mathbb{E}[\mathcal{R}(\mathcal{K})]$ can be interpreted as the following linear program:

$$\begin{aligned}&\min \sum_i \sum_a \sigma_{i,a} \mathcal{R}(Y_i^n, a_i^n) \\ &\text{subject to } \sigma_{i,a} \geq 0, \quad \forall i, a, \\ &\sum_i \sum_a \sigma_{i,a} = 1, \quad \forall i, a, \\ &\sum_a \sigma_{ja} = \sum_i \sum_a \sigma_{i,a} p_{i,j}, \quad \forall i, a, j,\end{aligned} \tag{57}$$

$\mathbb{E}[\mathcal{R}(\mathcal{K})]$ is a special case of the linear programm and is solved by a standard linear linear programming algorithm known as *the simplex algorithm*. The simplex algorithm solves the linear program by moving from an extreme point of the feasibility region to a better extreme point until the optimal is reached. So we can figure out the lower bound and upper bound of the lifetime by the linear programming (57). □

Lemma 10. *Under the EC-CKN algorithm, the upper bound network lifetime is*

$$\underline{\mathcal{L}}_{\text{EC-CKN}} = \frac{\mathcal{E}_0}{\max \sum_i \sum_a \sigma_{i,a} \mathcal{E}_{\text{epoch}}^n}, \tag{58}$$

where $\sigma_{i,a}$ is the steady-state probability that the action a is chosen when the chain is in the state i under the policy $\mathcal{K}$.

7. Simulation

Simulation Setup. In NetTopo [10], we conduct extensive simulation experiments. The studied WSN has the network size $800 \times 600\ \text{m}^2$. The number of deployed sensor nodes are increased from 100 to 1000 (each time increased by 100). The value of k is changed from 1 to 10 (each time increased by 1). For every number of deployed sensor nodes, we use 100 different seeds to generate 100 different network deployment. A source node is deployed at the location of (50, 50), and a sink node is deployed at the location of (750, 550). The transmission radius for each node is 60 m.

Routing Algorithm. TPGF routing algorithm [11] is one of the earliest geographical multipath routing algorithms designed for facilitating the multimedia stream data transmission in static and always-on wireless sensor networks (WSNs). It focuses on exploring the maximum number of optimal node-disjoint routing paths in network layer in terms of minimizing the path length and the end-to-end transmission delay. TPGF routing algorithm includes two phases. Phase 1 is responsible for exploring the possible routing path. Phase 2 is responsible for optimizing the found routing path with the least number of hops. The simulation results with changed k values in this figure reflect the comparison between the original average length of paths and the optimized average length of paths.

Sleep Probability of the CKN Algorithm. The node's sleep probability has been analyzed in Section 4.1. Figure 5 describes the node's theoretic sleep probability based on the probability model that has been set up and covers the comparison between the simulation results and the theoretic value. We enlarge the some factors of the probabilistic model so that the theoretic value is greater than the simulation when $N/k > 100$. While $N/k < 100$, the theoretical results and simulation results approximate every much. Moreover, the same variation trend proves that our model is ponderable for nodes' sleep probability in the CKN algorithm.

Network Lifetime Under the CKN Algorithm. Based on the node sleep probability from the probabilistic model and simulation, we can get the relative probability stretch variation curve with the epoch. *Relative probability stretch* is defined as a function of the expected number of epochs with k active neighbors compared to the expected number of epochs with a larger $|N_u|$ active neighbors. In our work, we assume that the radio dissipates $E_{\text{elec}} = 50\,\text{nJ/bit}$ to the transmitter or receiver circuity and $\epsilon = 100\,\text{pJ/bit/m}^2$, and the data rate is 20 kbps. In Figure 6, there is a key value of the length of the epoch time ($M \cdot t$) when N and k are certain, which is the intersection of relative probability stretch and reference axis. The CKN algorithm is energy efficient if the epoch is less than the key value. Otherwise, the node will consume more energy by the CKN algorithm than it is always active.

Network Lifetime Comparison between the CKN Algorithm and the EC-CKN Algorithm. The network lifetime of the CKN algorithm and the EC-CKN algorithm in a WSN is represented by the number of epochs. We conduct simulation for the CKN algorithm and the EC-CKN algorithm in a WSN and compare the network lifetime under the same situation in Figures 7 and 8. Results in Figures 7(a) and 7(c) confirm that the energy consumption of the EC-CKN algorithm is better managed and balanced than the CKN algorithm. Results in Figures 7(b) and 7(d) reveal the influence of changing the value of k: decreasing the value of k in the EC-CKN algorithm can prolong the network lifetime, particularly when the network nodes are densely deployed. Figures 7(a) and 7(b) show the whole distribution and tendency varying from the different combinations between k and N (the total number of deployed nodes in the network). Correspondingly, Figures 7(c) and 7(d) have the same meaning for the EC-CKN algorithm. Figure 8 compares the network lifetime between the CKN algorithm and the EC-CKN algorithm, by conducting the two algorithms under the same scenario. And the results give a straight proof that the

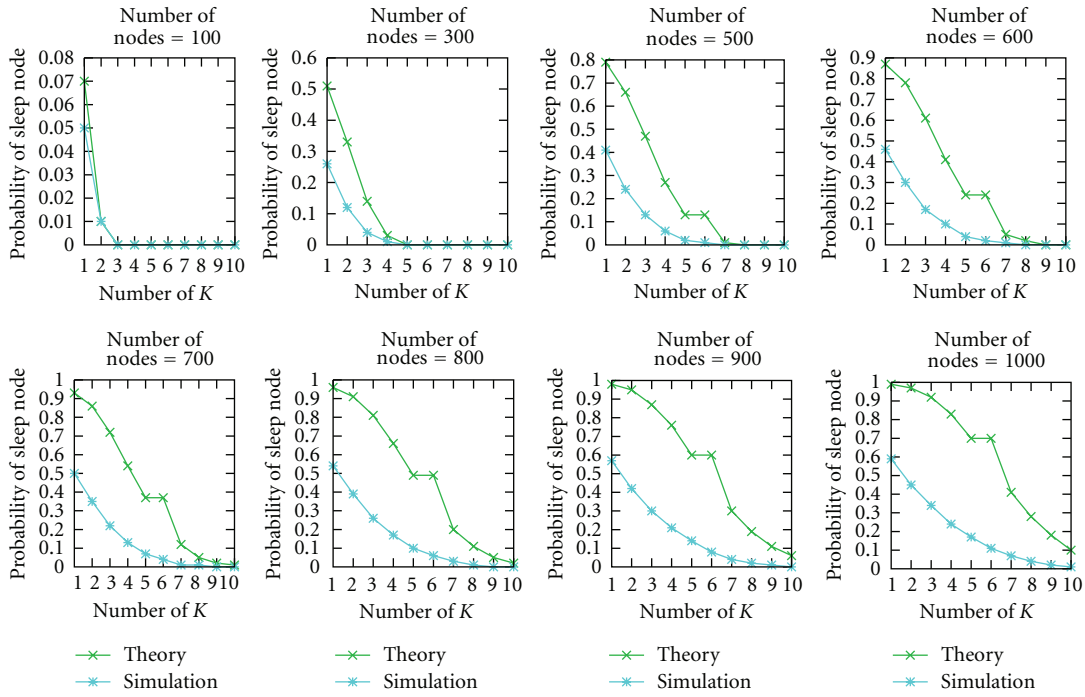


Figure 5: Node's sleep probability. The theoretical results and simulation results do not match well when the network size gets bigger, such as $N = 700$ 1000. The reason is that we enlarge the probability that any two nodes in C_u are connected either directly themselves or indirectly through s_u's 2-hop neighbors.

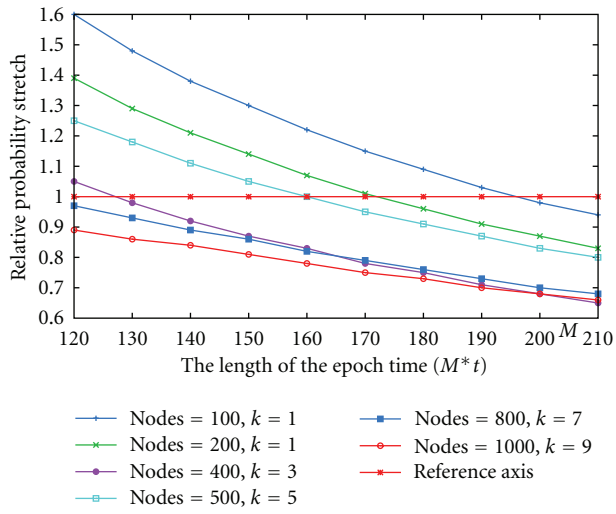
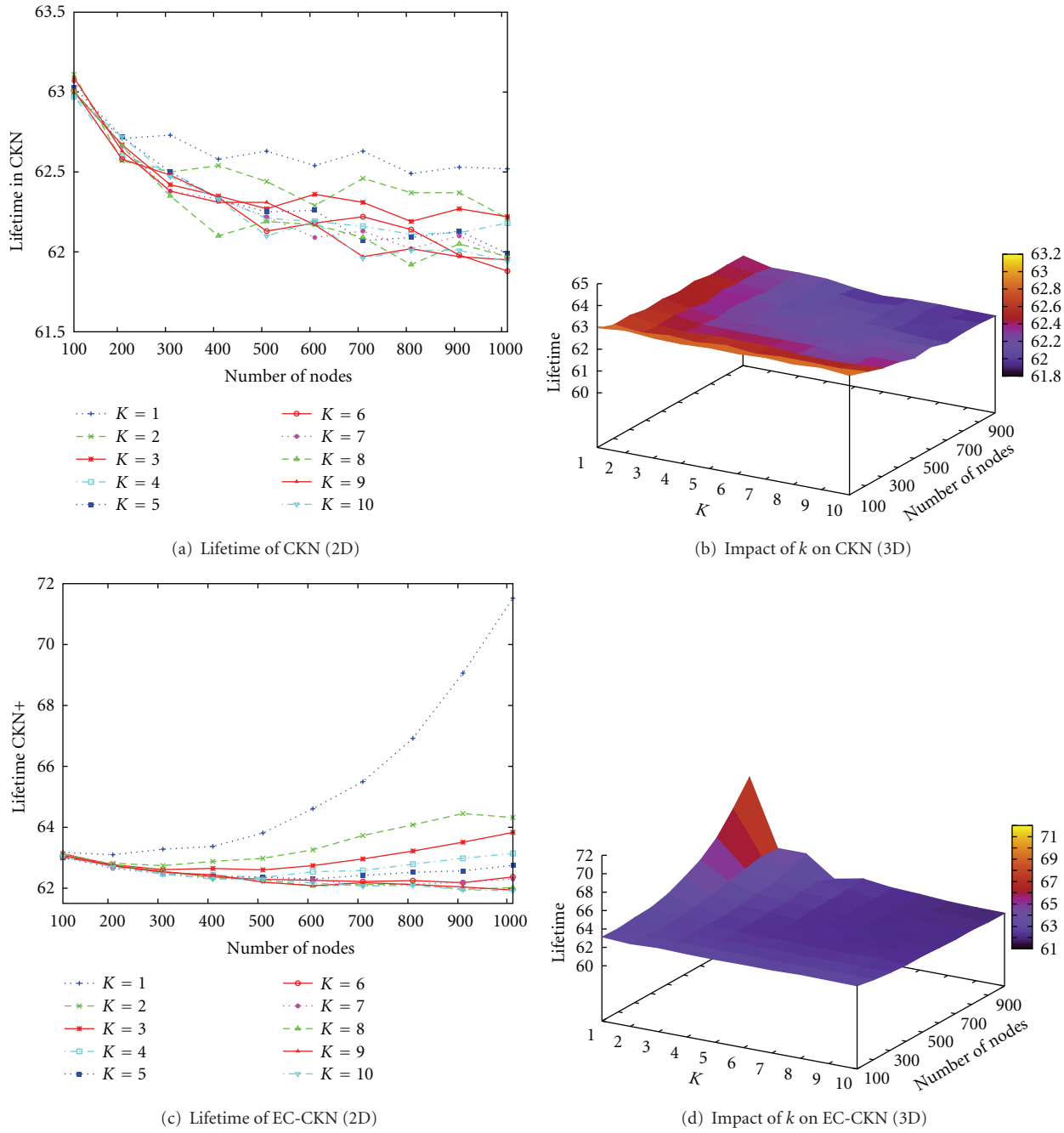


Figure 6: Relative probability stretch varies with epoch.

EC-CKN algorithm can provide longer network lifetime than the CKN algorithm.

8. Related Work

Network lifetime has been defined in various ways [12–17], and an energy-efficient mechanism may choose to maximize a certain type of network lifetime. One useful mechanism is the MAC layer power saving scheme, which reduces energy consumption by minimizing radio transceivers' idle time. SMAC [18] is an important MAC protocol designed for sensor networks, which forces sensor nodes to operate at low duty cycle by putting them into periodic sleep instead of idle listening. The timeout-MAC protocol (TMAC) improves SMAC by using an adaptive duty cycle [19]. Data-gathering MAC (DMAC) also uses an adaptive duty cycle, which provides low node-to-sink latency in convergecast communication by staggering the wake-up times of the nodes in the convergecast tree [20]. Pattern MAC (PMAC)



(a) Lifetime of CKN (2D)

(b) Impact of k on CKN (3D)

(c) Lifetime of EC-CKN (2D)

(d) Impact of k on EC-CKN (3D)

FIGURE 7: The network lifetime of running the CKN algorithm and the EC-CKN algorithm, respectively. In (a), the ten lines of the lifetime results according to the CKN algorithm have a lot intersections, which means the energy consumption by the CKN algorithm in a WSN is not managed towards the energy-balancing direction. However, in (c), the ten lines of the lifetime results by the EC-CKN algorithm present smooth changing when the number of nodes and the value of k are changed and the lifetime increases when the ratio N/k increases. This point clearly reflects that the energy consumption by the EC-CKN algorithm in a WSN is well managed towards the energy-balancing direction. Furthermore, simulation results in (b) and (d) also reveal that decreasing the value of k (let more nodes sleep) can definitely help to prolong the network lifetime of the EC-CKN algorithm in a WSN, but not the CKN algorithm.

[21] allows each sensor node determines the sleep-wake-up schedules based on its own traffic and the traffic patterns of its neighbors. BMAC [22] and XMAC [23] are two asynchronous duty-cycle-based protocols. In BMAC, each sensor node periodically wakes up to check whether there is any activity currently on the wireless channel or not. If so, the node remains active to receive a possible incoming packet. In this way, the node will receive one or more packets that are actually destined for other nodes. XMAC uses a strobed preamble to solve the overhearing

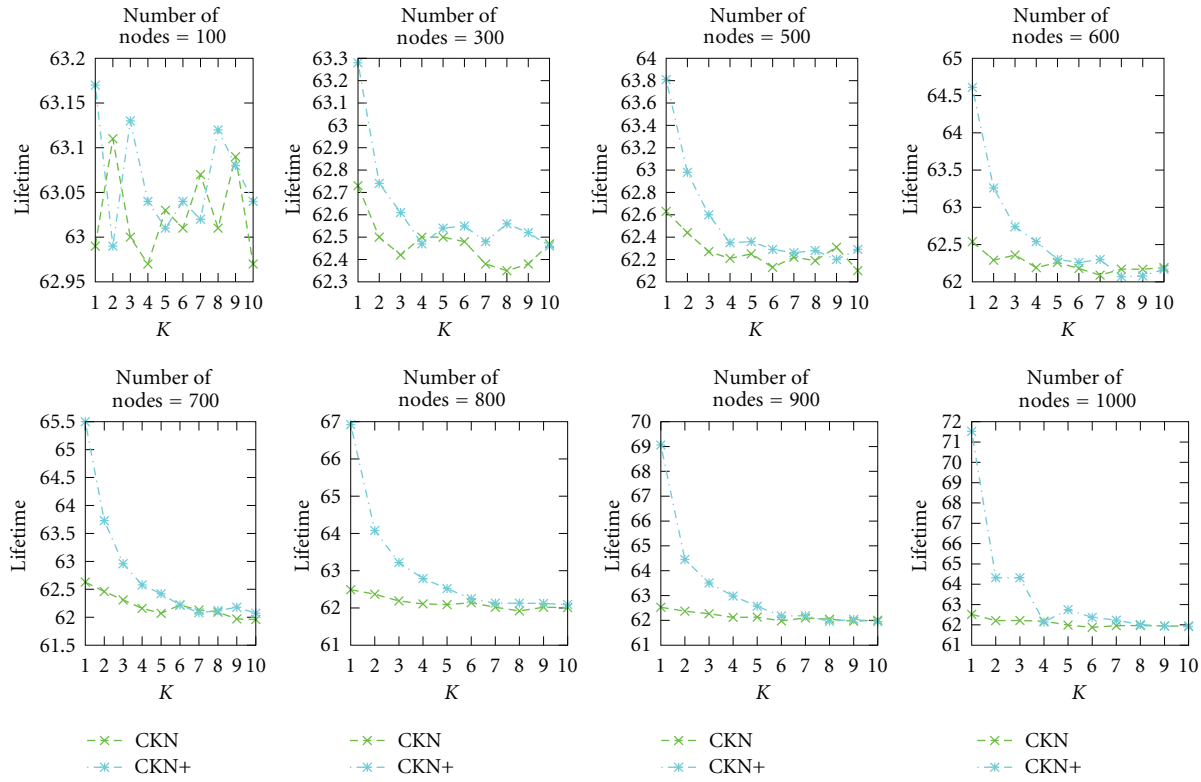


Figure 8: Network lifetime comparison between the CKN algorithm and the EC-CKN algorithm. Y-axle represents the recorded number of epochs when the first node runs out of its energy.

problem in BMAC. A strobed preamble includes a sequence of short preambles prior to DATA transmission. Obviously, the preamble transmission of BMAC and XMAC occupies the wireless medium for a long time under the light traffic load. To solve the problem, WiseMAC [24] is proposed. WiseMAC is similar to BMAC except that senders efficiently reduce the length of the wake-up preamble by exploiting the sampling of the schedules of its neighbors.

Another common technique to minimize the energy consumption and extend the network lifetime is to put some sensors in the sleep state and put others in the active state for the sensing and communication tasks. When a sensor is in the sleep state, its processor is turned off, but a timer or some other triggering mechanism may be running to wake up the sensor. On the other hand, all the components in the sensor are turned on when it is in the active state. Therefore, the energy consumed in the sleep state is only a tiny fraction of that consumed in the active state. One complexity here is that different types of sensors may support different sets of states. For example, the μAMPS sensor has three sleep states: *Monitor*, *Observe*, and *Deep Sleep* [25]. A sleep scheduling mechanism allows each sensor to determine when it should switch its state and what state it will switch to. Kumar et al. adopt the randomized independent scheduling (RIS) mechanism extending network lifetime while achieving asymptotic K-coverage [26]. RIS assumes that time is divided into epochs based on a time synchronization method. At the beginning of a epoch, each sensor independently decides weather to become active with probability p or go to sleep with probability $1 - p$. Thus, the network lifetime is increased by a factor close to $1/p$. Berman et al. presented a centralized and a distributed algorithm to maximize network lifetime while achieving K-coverage [27]. In the distributed algorithm, each sensor is in one of three states: *active*, *idle*, or *vulnerable*. In the vulnerable state, if the sensor discovers that part of its sensing area cannot be covered by any of its active or vulnerable neighbors, it immediately enters the active state. Otherwise, it enters the idle state if its sensing area can be monitored by either active neighbors or vulnerable neighbors with a higher energy level. Another distributed scheduling mechanism named lightweight deployment aware scheduling (LDAS) is proposed in [16]. Unlike the aforementioned distributed algorithm, LDAS does not ask if the sensor nodes are equipped with GPS or other devices to obtain location information. It assumes that each active node knows the number of its active neighbors. If the number of the active nodes exceeds a threshold, the node randomly selects some of its neighbors and sends tickets to them. When a node receives enough tickets from its neighbors, it may enter the sleep state after a random backoff period. Probing environment and adaptive sensing (PEAS) mechanism was designed for high-density sensor networks in a harsh environment [17]. Each node broadcasts a message with a transmission range of t_r after sleeping for a random

period. A node will go to the active state only if it receives no replies from its active neighbors. PEAS assumes that sensor nodes may fail frequently and unexpectedly, which makes synchronized sleeping algorithm infeasible. Coverage configuration protocol (CCP) is an integrated coverage and connectivity configuration protocol [4]. The protocol defines that nodes have three states: ACTIVE, LISTEN, and SLEEP. Each node is initially ACTIVE. When it receives a message, it goes to the LISTEN state and starts a random LISTEN timer. The node will go to SLEEP if it satisfies the following conditions: (i) its LISTEN timer expires; (ii) the network is still connected when it goes to SLEEP. In the SLEEP state, a node will set a random SLEEP timer. When the timer expires, it will enter the LISTEN state. PECAS (probing environment and collaborating adaptive sleeping) is an extension of PEAS. A active node in PECAS will go back to sleep after a specified period of time. It piggybacks the remaining ACTIVE time in its reply messages to its neighbors' probe message. Therefore, an active neighbor who will go to SLEEP can schedule itself to wake up before the node goes to SLEEP, which prevents the occurrence of blind spots. Adaptive self-configuring sensor networks topologies (ASCENT) is similar to PEAS, which is also designed for high-density sensor networks. However, unlike PEAS, ASCENT does not guarantee network connectivity. Low-energy adaptive clustering hierarchy (LEACH) [28] is a cluster-based protocol, which utilizes randomized rotation of cluster heads to distribute work load among the sensors. In LEACH, the lifetime of network is divided into epochs, and each epoch includes a *set-up* phase and a *steady* phase. During the set-up phase, cluster heads are selected and each sensor joins a cluster by choosing a cluster that needs the minimum communication energy. During the steady phase, each cluster head aggregates the data from the sensors in its cluster and then forwards them to the sink. To conserve energy, nonhead sensors go to sleep at all time except that they are transmitting data. E-LEACH [15] is an extension of LEACH, which adapts the cluster heads selection algorithm to the nonuniform starting energy level among the sensors and the required number of cluster heads.

9. Conclusion

When deploying real WSNs for practical applications, it is extremely important to have a good sleeping scheduling algorithm to balance sensor nodes' energy consumption and a reasonable length for an epoch towards energy saving is extremely important. In this paper, we make the following major contributions for supporting the real WSNs applications. (1) A theoretical study is given for the CKN algorithm, which formulates the lower bound of an epoch to keep the CKN algorithm feasible. (2) A new sleep scheduling algorithm, named as EC-CKN, is proposed to balance the energy consumption and prolong the network lifetime. (3) Extensive simulation work is conducted, which proved the energy consumption in the EC-CKN algorithm is well balanced.

Acknowledgments

This work is partially supported by Natural Science Foundation of China under Grant no. 61070181 and Natural Science Foundation of Liaoning Province under Grant no. 20102021. L. Shu's research in this paper was supported by Grant-in-Aid for Scientific Research (S)(21220002) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] S. Nath and P. B. Gibbons, "Communicating via fireflies: geographic routing on duty-cycled sensors," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 440–449, April 2007.

[2] C. F. Hsin and M. Liu, "Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 433–442, April 2004.

[3] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 20–27, April 2005.

[4] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenS '03)*, pp. 28–39, November 2003.

[5] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design and Test of Computers*, vol. 18, no. 2, pp. 62–74, 2001.

[6] I. F. Akyildiz, M. C. Vuran, and O. B. Akan, "A cross-layer protocol for wireless sensor networks," in *Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS '06)*, pp. 1102–1107, March 2006.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences (HICSS '00)*, p. 223, January 2000.

[8] Y. Chen and Q. Zhao, "On the lifetime of wireless sensor networks," *IEEE Communications Letters*, vol. 9, no. 11, pp. 976–978, 2005.

[9] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '02)*, pp. 80–91, June 2002.

[10] L. Shu, C. Wu, Y. Zhang, J. Chen, L. Wang, and M. Hauswirth, "NetTopo: beyond simulator and visualizer for wireless sensor networks," in *Proceedings of the 2nd International Conference on Future Generation Communication and Networking (FGCN '08)*, pp. 17–20, December 2008.

[11] L. Shu, Z. Zhou, M. Hauswirth, D. Le Phuoc, P. Yu, and L. Zhang, "Transmitting streaming data in wireless multimedia sensor networks with holes," in *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia (MUM '07)*, pp. 24–33, Oulu, Finland, December 2007.

[12] A. Cerpa and D. Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," in *Proceedings of the IEEE Computer and Communications Societies (INFOCOM '02)*, pp. 1278–1287, June 2002.

[13] J. Deng, Y. S. Han, W. B. Heinzelman, and P. K. Varshney, "Scheduling sleeping nodes in high density cluster-based sensor networks," *Mobile Networks and Applications*, vol. 10, no. 6, pp. 825–835, 2005.

[14] T. He, S. Krishnamurthy, J. A. Stankovic et al., "Energy-efficient surveillance system using wireless sensor networks," in *Proceedings of the 2nd International Conference on Mobile Systems, Applications and Services (MobiSys '04)*, pp. 270–283, 2004.

[15] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[16] K. Wu, Y. Gao, F. Li, and Y. Xiao, "Lightweight networks," *Mobile Networks and Applications*, vol. 10, no. 6, pp. 837–852, 2005.

[17] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang, "PEAS: a robust energy conserving protocol for long-lived sensor networks," in *Proceedings of the 23th IEEE International Conference on Distributed Computing Systems (ICDCS '03)*, pp. 28–37, May 2003.

[18] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the IEEE Computer and Communications Societies (INFOCOM '02)*, pp. 1567–1576, New York, NY, USA, June 2002.

[19] T. Van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 171–180, Los Angeles, Calif, USA, November 2003.

[20] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Proceedings of 18th International Parallel and Distributed Processing Symposium (IPDPS '04)*, pp. 3091–3098, April 2004.

[21] T. Zheng, S. Radhakrishnan, and V. Sarangan, "PMAC: an adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*, p. 237, April 2005.

[22] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 95–107, November 2004.

[23] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys' 06)*, pp. 307–320, November 2006.

[24] A. El-Hoiydi and J. D. Decotignie, "WiseMAC: an ultra low power MAC protocol for multi-hop wireless sensor networks," in *Proceedings of the 1st International Workshop on Algorithm Aspects of Wireless Sensor Networks*, vol. 3121 of *Lecture Notes in Computer Science*, pp. 18–31, 2004.

[25] E. Shih, S. H. Cho, N. Ickes et al., "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM '01)*, pp. 272–286, July 2001.

[26] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MOBICOM '04)*, pp. 144–158, October 2004.

[27] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '04)*, pp. 2329–2334, 2004.

[28] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences (HICSS '00)*, p. 223, January 2000.

[29] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design and Test of Computers*, vol. 18, no. 2, pp. 62–74, 2001.