CREEC: Chain Routing with Even Energy Consumption

Jisoo Shin and Changjin Suh

Abstract: **A convergecast is a popular routing scheme in wireless sensor networks (WSNs) in which every sensor node periodically forwards measured data along configured routing paths to a base station (BS). Prolonging lifetimes in energy-limited WSNs is an important issue because the lifetime of a WSN influences on its quality and price. Low-energy adaptive clustering hierarchy (LEACH) was the first attempt at solving this lifetime problem in convergecast WSNs, and it was followed by other solutions including power efficient gathering in sensor information systems (PEGASIS) and power efficient data gathering and aggregation protocol (PEDAP).**

Our *solution—chain routing with even energy consumption* (CREEC)—solves this problem by achieving longer average lifetimes using two strategies: i) Maximizing the fairness of energy distribution at every sensor node and ii) running a feedback mechanism that utilizes a preliminary simulation of energy consumption to save energy for depleted sensor nodes.

Simulation results confirm that CREEC outperforms all previous solutions such as LEACH, PEGASIS, PEDAP, and PEDAP-power aware (PA) with respect to the first node death and the average lifetime. CREEC performs very well at all WSN sizes, BS distances and battery capacities with an increased convergecast delay.

Index Terms: **Chain, chain routing with even energy consumption (CREEC), fair energy consumption, Kruskal's minimum spanning tree (MST), low-energy adaptive clustering hierarchy (LEACH), link swap, wireless sensor networks (WSNs).**

I. INTRODUCTION

Wireless sensor networks (WSNs) use distributed sensor nodes to monitor various conditions of remote locations such as temperature, sound, vibration, pressure, motion and pollutants. A WSN is configured autonomously by sensor nodes equipped with sensing, computing and wireless communication capabilities [1]–[3]. WSNs have many variants depending on applications and environments [4]–[6].

Maintaining a long lifetime is important to overcome the limited and non-refilled battery equipped in sensor nodes. Heinzelman *et al.* [7] proposed low-energy adaptive clustering hierarchy (LEACH) that performs the "full-fusion convergecast" routing to extend WSN lifetime.

We would like to introduce our WSN model and related terminologies. A WSN is assumed to have a remote base station(BS) as shown in Fig. 1. A BS is an intelligent gateway node to the outside of a WSN and is accessed by WSN operators. A BS is equipped with unlimited electricity and high computation power. Sensor nodes can adjust transmission power according to their transmission radius. In this paper, a node means a sensor node excluding a BS.

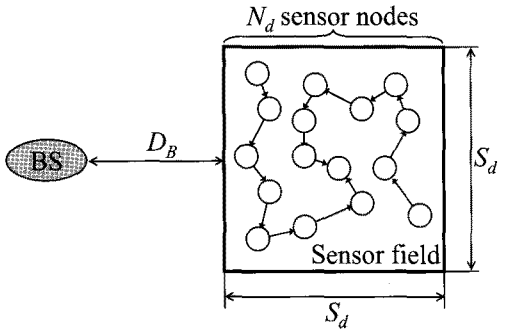


Fig. 1. Description of a WSN.

A convergecast is the routing that every sensor node periodically forwards measured data to the BS along configured routing paths. Full-fusion or perfect fusion means that many b-bit packets can be compressed into a single b-bit packet. The full-fusion is used to collect maximum, minimum, total and average values of sensed data. Due to its simpleness, the full-fusion routing has attracted many researcher's interest [7]–[9]. A general fusion-rate WSN routing model [10] can be built with properly combining a few full-fusion WSN routing solutions.

To save energy in a convergecast, every node transmits a single packet along the routing path after merging all received packets and its own measured data. Fig. 2 introduces popular types of the routing topology in WSNs–a hierarchical topology, a spanning tree, and a chain in Figs. 2(a), 2(b), and 2(c), respectively. In these figures, each routing path connects 17 small circles representing 17 sensor nodes in a square sensor field below the BS. All packets are forwarded to the BS along the directed links in Fig. 2.

We call 'throwing' which is transmitting a packet directly to the BS. A throwing node is represented by a dotted circle in Fig. 2. Four throwing nodes appear in Fig. 2(a), which are cluster heads in charge of each cluster. Note the throwing distances to the BS outside the sensor field are very long. To get over long throwing distances, a full-fusion WSN prefers to have a single throwing node.

Chain routing with even energy consumption (CREEC) achieves a longer average lifetime with a few strategies using centralized control at the BS. The BS calculates the routing paths and schedules throwings. In CREEC, the BS has important roles to predict and simulate energy consumption at every node. As well, CREEC uses a strategy to maintain two rules. They are i) to maximize the fairness of energy distribution, and ii) to minimize the total energy consumption if energy minimization does not damage the rule i). CREEC also uses a feedback mechanism of energy distribution. The BS calculates the

Manuscript received June 11, 2009; approved for publication by Abbas Jamalipour, Division II Editor, February 28, 2010.

This work was supported by the Soongsil University Research Fund(2010).

The authors are with the Department of Computing, Soongsil University, 511 Sangdo-Dong, Dongjak-Gu, Seoul, 156-743 Korea, email: jsshin@netwarking.com, cjsuh@ssu.ac.kr.

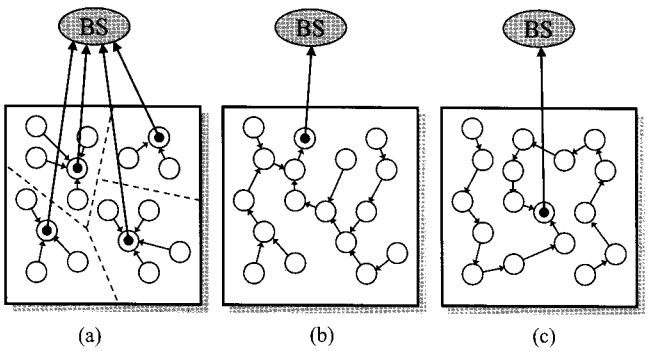


Fig. 2. WSN routing topologies: (a) A hierarchical topology (LEACH, HPEGASIS,COSEN, CMST-DC), (b) a tree (PEDAP, PEDAP-PA), (c) a chain (PEGASIS, CREEC).

consumed energy at all nodes, and the energy distribution is used when re-building a chain so that the BS assigns light roles to energy-depleted nodes and heavy jobs to energy-sufficient nodes.

To implement these strategies, we propose the following tactics. CREEC separately defines two types of transmission—throwing directly sent to the BS and 'forwarding' to a neighbor node. CREEC compels all nodes to spend the same amount of throwing energy E_{th} and forwarding energy E_{fw}. The fair throwing energy distribution is achieved by assigning the number of throwing inversely proportional to the one-time throwing energy at each node.

We also develop an enhanced chain algorithm that produces shorter lengths and fairer energy consumption. We define a node degree as the number of neighbor nodes in the established spanning tree or chain. In a chain, all non-leaf nodes have the node degree of 2. A chain is longer than the minimum spanning tree (MST), but is easier for energy control due to the strict and uniform node degree constraint.

The chains generated by CREEC are so excellent that many chain based routing schemes [9], [11], [12] become better by simply substituting our chain. Our chain algorithm works good too in case when chains are used for other purposes - for automatically dividing all nodes into clusters [9], [12].

The paper is organized as follows. Section II introduces related researches. Section III defines WSN modeling and assumption. Section IV describes and explains the CREEC algorithm and operation. Section V presents and analyzes our simulation results compared with competing routing schemes. Lastly Section VI concludes our work.

II. RELATED RESEARCHES

This section introduces publications about the full-fusion convergecast WSN routing. Fig. 2 shows the classification of routing schemes. A full-fusion convergecast problem can be classified according to the use of hierarchy. Hierarchical routing in Fig. 2(a) is proposed to reduce the routing calculation complexity and the delay for a convergecast. A hierarchical WSN has three basic node levels. They are non-cluster head nodes, cluster heads and the BS. A cluster head is in charge of a cluster. The cluster head collects all data in its cluster and sends the merged data to the BS. To consider that a cluster head consumes much energy for the cluster management, cluster heads have to be updated frequently. And so do the routing paths. In non-hierarchical routing, on the other hand, a routing path is used for a long time. A tree topology and a chain topology are used as shown in Figs. 2(b) and 2(c), respectively. We skip the star topology which is sometimes used for simple but cheap products.

Let us introduce a few hierarchical routing algorithms. Heinzelman *et al.* [7] proposed a full-fusion convergecast WSN lifetime problem after modeling a WSN, and announced LEACH as a solution. LEACH uses a three-level hierarchical routing including the BS level, organizing a two-stage star topology in Fig. 2(a). Every non-cluster head node in LEACH sends measured data to the BS via the selected cluster heads. Dividing into clusters and choosing cluster heads are determined distributively in the first version [7] and centralized at the BS in later one [13]. We call a 'round' of convergecast collecting measured data from all nodes to the BS. LEACH uses 'super rounds'. When a new super round begins, cluster heads are updated and new paths via the new cluster heads are constructed. Later, LEACH's variants showed up depending how to cover a cluster. A chain is used in chain oriented sensor network for efficient data collection (COSEN) [11], and a tree is used in cluster-based minimal spanning tree with degree-constrained (CMST-DC) [14].

Lindsey *et al.* [9] enhance the hierarchical routing from LEACH in two ways—hierarchical power efficient gathering in sensor information systems (HPEGASIS)1 and HPEGASIS2. They proposed an idea to use a chain in dividing a sensor field into a few virtual clusters. HPEGASIS1 builds a tree of $(\lceil \log_2 N_d \rceil + 1)$-level hierarchy including the BS level in an N_d-node WSN. Assume there are 16 nodes in a WSN and they are termed as $1, 2, 3, \cdots, 16$ along the chain. At each level, grouping is made such as $\{(1,2),(3,4),\cdots,(15,16)\}$ at first and $\{((1,2),(3,4)),\cdots,((13,14),(15,16))\},\cdots$. And a head is selected at every hierarchical group. In HPEGASIS1 the nodes $1, 2, \cdots, 16$ sequentially perform throwing. HPEGASIS1 proposes a simple rule for a given throwing node how to select a head at every level.

HPEGASIS2 generates a 4-level hierarchy compared to the 3-level in LEACH. A single master cluster head adds an extra level. It collects every cluster head's data and performs throwing to the BS. HPEGASIS2 is a generalized version of HPEGASIS1 in that a group size is relieved from 2 to a general one. It is reported that HPEGASIS2 can be improved by selecting the cluster head and master cluster head as the node whose residual energy is the largest [12].

The next routing category is a tree topology without a hierarchy. Power efficient data gathering and aggregation protocol (PEDAP) [8] uses the MSTs as in Fig. 2(b) to minimize the total transmission energy. In PEDAP, the nearest node to the BS becomes a throwing node and maintain throwing until it dies. This rule results in frequent reconfigurations due to the early death of throwing nodes, because throwing operation requires heavy transmission energy.

In PEDAP, the BS calculates the MST in the weighed graph in which every wireless link $l(k,j)$ with a distance $d(k,j)$ has the link weight $C(k,j)$ according to (1). $C(k,j)$ indicates the

transceiving energy of a b-bit packet through $l(k,j)$. B in (1) means the BS. (1) includes two physical constants E_{elec} and E_{amp}.

$$C(k,j) = \begin{cases} 2E_{\text{elec}}b + E_{\text{amp}}bd^2(k,j) & \text{if } j \neq B, \\ E_{\text{elec}}b + E_{\text{amp}}bd^2(k,B) & \text{if } j = B. \end{cases} \quad (1)$$

PEDAP has a power-awareness option called PEDAP-power aware (PA). PEDAP-PA tries to achieve fairer energy consumption at every node by sacrificing the minimum tree length. In PEDAP-PA, the BS is aware of the current energy level e_k at every node k. PEDAP-PA modifies the given graph by dividing the right expression in (1) by e_k. The inserted $1/e_k$ term helps energy saving in relatively depleted nodes and thus contributes fairer energy consumption. PEDAP-PA unfortunately uses distorted link weights that do not represent the real energy consumption. So the generated trees are larger than the MST. Both PEDAP and PEDAP-PA have seriously unfair energy distribution at every node.

A chain topology in Fig. 2(c) is the last routing category. Power efficient gathering in sensor information systems (PEGASIS) [9] and our CREEC belong to this category. PEGASIS generates a chain by using slightly modified Prim's MST algorithm [15] to keep the node degree 2 at a non-leaf node. In PEGASIS, throwings are accomplished fairly in accordance with the order of nodes along the chain. PEGASIS chains are used to have long links especially at a few of last choices. It is because of the limitation of greedy algorithm that PEGASIS's chains are not allowed to revoke the already selected links. A non-greedy MEDC algorithm [12] generates shorter chains than PEGASIS chains.

III. WSN MODELING

This section describes modeling of convergecast WSNs. It deals with WSN environments, sensor nodes and energy consumption mostly cited from [7]. We recommend to be acquainted with notations in Fig. 1 in advance.

- WSN: A WSN consists of the N_d number of non-mobile nodes and a fixed BS. Nodes are distributed randomly within a square WSN field. The BS is located away from the WSN field.
- BS: The BS has powerful CPUs, enough memory and a rechargeable battery. It collects every node's measured data and manages a WSN. It may generate routing trees, and announce to every node all necessary routing hops for convergecasts.
- Nodes: Each node is homogeneously equipped with sensors, an intensity-controlled transmitter, a receiver and a non-refilled battery. A node measures transmission and reception energy exactly by using modern wireless transmission technologies [16].
- Node lifetime: Nodes begin with the same initial energy level E_0. Completely depleted nodes stop operation and are removed from the routing tree. A new reconfigured tree is generated automatically.
- Perfect fusion: Each node generates and sends a single b-bit packet. A forwarding node compresses all received b-bit packets and its own sensing data into a b-bit packet.
- Energy modeling: A node consumes energy E_t and E_r for transmitting and receiving a b-bit packet respectively as shown in (2) and (3) [7]. d means transmission distance. We use E_{elec} as 50 nJ/bit, E_{amp} as 0.1 nJ/bit/m^2, and the packet length b as 2000 bits. We ignore energy for reconfiguring routing trees, sensing and data fusion.

$$E_t(d^2) = E_{\text{elec}}b + E_{\text{amp}}bd^2 \quad \text{and} \quad (2)$$

$$E_r = E_{\text{elec}}b. \quad (3)$$

Some routing schemes may require extra functions to achieve better performance. We list those functions and names in parentheses.

- A node can measure the residual battery level exactly (PEDAP-PA).
- The BS calculates and stores transceiving energy spent at every node and foresees the future energy level through a simulation. (CREEC)

IV. CREEC

This section defines and explains CREEC. Subsection IV-A summarizes basic operations and introduces strategies and operation required. CREEC mathematically defines forwarding and throwing separately. Subsection IV-B deals with calculation of throwing energy and assignment of throwing for every super round. Subsection IV-C explains how distribution of forwarding energy at every node becomes balanced. Because forwarding energy heavily relies on the shape of chains, we explain how to build them.

A. Overview

CREEC runs well with partial network information. Before introducing the CREEC's operation, we give a comment on it. At the beginning, the BS orders every node k to report i) the throwing energy $E_t(d^2(k,B))$ from k to BS, ii) the adjacency list $A_{dj}(k)$ that includes the a_{dj} number of the closest neighbor nodes at k and iii) transmission energy $E_t(d^2(k,j))$ from k to any neighbor node j in $A_{dj}(k)$.

A temporary cost $e_x\hat{c}(i,j)$ is used for an un-reported link $l(i,j)$. e_x is an arbitrary constant having a large value. Say, it is 10. The BS calculates the calculated costs $\hat{c}(i,j)$ by applying the all-to-all shortest path algorithm at a given network built with ii) and iii) for every k. The algorithm, for example the Floyd-Warshall all-to-all shortest path algorithm, solves every shortest distances $D(*,*)$. $D(i,j)$ is used as the calculated cost $\hat{c}(i,j)$ for the un-reported link $l(i,j)$. Because temporary costs are exaggerated by e_x times, they are replaced one by one in the process of constructing a convergecast tree. Finally, the convergecast path only includes the reported links.

The following is a brief overview of CREEC's main operation. To prepare a new super round, the BS performs a few tasks. The BS firstly calculates the number of throwing $n_{th}(k)$ for a super round at each node k, and then creates a chain that reflects the lifetime cumulative forwarding energy spent. The BS then broadcasts the newly established chain and the throwing schedule.

A round begins with the BS's broadcast. The BS requests a new convergecast and assigns a throwing node in every round. The convergecast starts at a leaf node in the chain. It transmits a packet to its neighbor node. The node that receives a packet from its neighbor node forwards it to the other neighbor node. This is repeated until the packet arrives at the throwing node. A non-leaf throwing node needs merging data from two sub-chains. To complete merging, the throwing node alerts the other leaf node, and the second half of convergecast is done similarly. The throwing node throws a packet to the BS after receiving packets from all neighbor nodes. To enable the parallel forwarding from two leaf nodes, the BS should provide the transmission schedule to avoid packet collision.

CREEC was designed on top of the philosophy to harmonize the next two incompatible rules.

- R_F : Let all nodes spend energy very fairly so that the distribution of energy consumption is as flat as possible.
- R_M : Let all nodes spend the minimum total energy.

R_F helps a WSN to maintain high sensing quality, and R_M extends WSN lifetimes. Most researches, especially [8], prefer R_M. However we noticed the opposite approach can be better.

CREEC tries to meet R_F first and to satisfy R_M conditionally. This approach has two advantages. First, prevention costs less than remedy. Energy waisted for seriously depleted nodes is generally larger than energy overhead for maintaining balanced energy distribution. Second, we can save reconfiguration energy. Suppose R_F is implemented very successfully that battery levels at all nodes are kept the same and become totally depleted simultaneously. Every node dies in a round and no tree repair is necessary. Repairing not only requires energy but also frequently causes malfunctions.

CREEC classifies transmission into throwing and forwarding, which are very different in nature. As the throwing distance is longer than the forwarding distance, a throwing node consumes more energy than non-throwing nodes. On the other hand, the total forwarding energy in a WSN is larger than the total throwing energy because throwing occurs at a throwing node for a round.

Based on the above analysis, CREEC succeeds in finding a way to meet R_F and R_M much better than the competing routing solutions do. In CREEC, the BS builds chains and schedules throwing so that all nodes spend the same throwing energy E_{th} and the same forwarding energy E_{fw}. The BS announces the chain information before a new super round begins. The BS may distribute throwing schedule to nodes both in super round basis or in round basis. The BS records the cumulative forwarding energy and the cumulative throwing energy at all nodes. CREEC can flatten the throwing energy distribution by controlling the number of throwing $n_{th}(\cdot)$ at every node. Super rounds are very useful in achieving R_F. The reconfigured spanning chains help nodes which previously spent much energy for forwarding choose shorter forwarding links to save forwarding energy, and this feedback flattens forwarding energy distribution.

CREEC uses chains. The best solution is the shortest chain that minimizes the sum of $C(k,j)$ given as $d^2(k,j)$ for a wireless link $l(k,j)$ in the established chain, and that spends the least forwarding energy. Finding the minimum chain is impractical because it is NP-complete. A chain is converted to a ring by putting a link that connects two leaf nodes. The shortest ring problem to find the length of the shortest ring is equivalent to the famous NP-complete traveling salesman problem. Thus we decided to circumvent the best solution and to find a locally optimum chain.

Now, we introduce how to enumerate the transceiving energy. $w(k)$ in (4) simulates the transceiving energy at a non-throwing node k. The node k whose node degree is $f(k)$ receives $(f(k)-1)$ packets from the child nodes and transmits a packet to its parent node j. (4) is converted to (5) by substituting E_t and E_r in (2) and (3).

$$w(k) = (f(k)-1)E_r + E_t(d^2(k,j)) \tag{4}$$

$$= b(E_{\text{elec}} f(k) + E_{\text{amp}} d^2(k,j)). \tag{5}$$

Let k_t be a throwing node and B be the BS. The throwing energy $v_{th}(k)$ is spent only at k_t as

$$v_{th}(k) = \begin{cases} 0 & \text{if } k \neq k_t, \\ E_t(d^2(k,B)) & \text{if } k = k_t. \end{cases} \tag{6}$$

We show how to enumerate the forwarding energy $u(k)$. A non-throwing node k spends all transceiving energy for forwarding. Thus $u(k)$ is same as $w(k)$ in (4). A throwing node k_t is required to receive $f(k_t)$ packets from $f(k_t)$ neighbor nodes before throwing. Listing up two cases makes (7). $p(k)$ in (7) represents k's parent node if k_t is regarded as the root node.

$$u(k) = \begin{cases} (f(k)-1)E_r + E_t(d^2(k,p(k))) & \text{if } k \neq k_t, \\ f(k)E_r & \text{if } k = k_t. \end{cases} \tag{7}$$

B. Throwing Schedule

This section calculates the number of throwing $n_{th}(k)$ to be assigned to any node k for a super round. Every node can unify the throwing energy, if the number of throwing is assigned as a real number in inverse-proportion to the one-time throwing energy. Unfortunately throwing is countable, and an integer value must be assigned for it. To overcome the discrepancy, we prepare an integer $n_{th}(k)$ and a real number $y_{th}(k)$.

We first calculate $y_{th}(k)$ in a N_d-node WSN. For simplicity we assume a super round to be 100 rounds long. Because throwing occurs once in a round, $y_{th}(k)$ becomes

$$\sum_{k=1,2,\cdots,N_d} y_{th}(k) = 100. \tag{8}$$

The expected throwing energy at a node in a super round - E_{th} - is defined and expressed alternatively using (6).

$$E_{th} = v_{th}(k) y_{th}(k)$$

$$= E_t(d^2(k,B)) y_{th}(k) \quad \text{for } k = 1,2,\cdots,N_d. \tag{9}$$

Replacing $y_{th}(k)$ in (8) with $\frac{E_{th}}{E_t(d^2(k,B))}$ from (9), E_{th} is calculated first and $y_{th}(k)$ is calculated accordingly

$$E_{th} = \frac{100}{\sum_{i=1,2,\cdots,N_d} \frac{1}{E_t(d^2(k,B))}}, \tag{10}$$

$$y_{th}(k) = \frac{100}{E_t(d^2(k,B))} \frac{1}{\sum_{i=1,2,\cdots,N_d} \frac{1}{E_t(d^2(i,B))}}. \quad (11)$$

To convert a real $y_{th}(k)$ to an integer $n_{th}(k)$, we need $y^0_{th}(k)$ and $y^-_{th}(k)$ to temporarily store roundoff errors in the current and the previous super rounds, respectively. Because $y_{th}(k)$ is independently determined, $\sum_{i=1}^{N_d} y_{th}(k)$ does not easily calculate to exactly 100, so a super round in CREEC is kept approximately 100 rounds long. $y^0_{th}(k)$ calculated at the $(s-1)$th super round is re-stored to $y^-_{th}(k)$ at the sth super round.

$$y^-_{th}(k) = \begin{cases} 0 & \text{for } s = 1, \\ y^0_{th}(k) & \text{for } s = 2, 3, \cdots. \end{cases} \quad (12)$$

We obtain $n_{th}(k)$ in (13) by rounding off the sum of $y_{th}(k)$ in (11) and $y^-_{th}(k)$ in (12) using the rounding off function $R_{nd}(\cdot)$. $n_{th}(k)$ in (13) is used to find the accumulated roundoff error $y^0_{th}(k)$ in (14).

$$n_{th}(k) = R_{nd}(y_{th}(k) + y^-_{th}(k)), \quad (13)$$

$$y^0_{th}(k) = y_{th}(k) + y^-_{th}(k) - n_{th}(k). \quad (14)$$

C. Chain Establishment

This subsection defines how to generate the locally minimal chains that consider energy saving for depleted nodes. The BS generates a chain with three steps in every super round. The first step classifies all nodes into three levels according to forwarding energy consumed until the super round is reached. The second step builds up a non-optimum greedy chain using the modified Kruskal's MST [17] algorithm. The last step reduces the chain length by repeatedly replacing a pair of long links with a shorter pair.

C.1 First Step

For even forwarding energy distribution, we need the cumulative forwarding energy $u_c(k, s-1)$ at a node k by accumulating the forwarding energy $u(k)$ until the $(s-1)$th super round in (15).

$$\begin{cases} u_c(k,-1) = u_c(k,\ 0) = 0. \\ u_c(k,\ s) \ = u_c(k, s-1) + u(k) \quad \text{for } s = 1, 2, \cdots. \end{cases} \quad (15)$$

The BS sorts all nodes according to the decreasing order of $u_c(*, s-1)$ and classifies them to three node levels. The two most depleted nodes in the sorted list belong to level-1, and the next $(R_{nd}(0.05N_d)-2)$ number of depleted nodes are selected as level-2. The rest are level-3 nodes.

The level-1 and level-2 nodes are given chances to make up previous energy losses. Level-1 nodes are assigned as two leaf nodes in the chain. Leaf nodes are characterized with the least node degree $f(k)(=1)$ and the less forwarding energy expressed in (7). A level-1 or level-2 node is given a chance to preoccupy a very short adjacent link. So their forwarding energy shrinks. In this way, forwarding energy consumption is compensated and balanced.

The BS predicts $u(k)$ which is the forwarding energy consumed in the upcoming super round s at a node k. Because $u(k)$ includes the throwing operation term $n_{th}(*)$ at (13), the throwing assignment has to precede chain establishment. Suppose there is a leaf node k, and k has a neighbor node k_n. For a super round s, k executes throwing for $n_{th}(k)$ times after receiving $n_{th}(k)$ packets from the neighbor k_n. k also sends $\sum_{i=1,\cdots,N_d, i\neq k} n_{th}(i)$ packets to k_n as a non-throwing leaf node. The upper case in (16) describes this situation.

$$u(k) = \begin{cases} \cdot \text{(for a leaf node } k) \\ \quad E_r n_{th}(k) + E_t(d^2(k,k_n)) \sum_{i=1,\cdots,N_d,\ i\neq k} n_{th}(i), \\ \cdot \text{(for a non-leaf node } k = \mu(j)) \\ \quad E_r(n_{th}(k) + \sum_{i=1,\cdots,N_d} n_{th}(i)) \\ \quad + E_t(d^2(k,k^-)) \sum_{i=1,2,\cdots,j-1} n_{th}(\mu(i)) \\ \quad + E_t(d^2(k,k^+)) \sum_{i=j+1,\cdots,N_d} n_{th}(\mu(i)). \end{cases} \quad (16)$$

Suppose k is not a leaf node. Starting at a leaf node, we assign the node position numbers $1, 2, \cdots, N_d$ one by one to the nodes along the chain. Let the nodes k^-, k and k^+ be assigned the node position numbers $(j-1)$, j and $(j+1)$ respectively. We use the permutation μ that converts a node position number to its node number. For examples, $\mu(j-1) = k^-$, $\mu(j) = k$ and $\mu(j+1) = k^+$. As a throwing node, the node k receives $2n_{th}(k)$ packets from its neighbor nodes k^+ and k^- spending $2n_{th}(k)E_r$. During the super round s, the node k can access the throwing node via the neighbor node k^- for $\sum_{i=1}^{j-1} n_{th}(\mu(i))$ times, and via the neighbor node k^+ for $\sum_{i=j+1}^{N_d} n_{th}(\mu(i))$ times after receiving $((\sum_{i=1}^{N_d} n_{th}(i)) - n_{th}(k))$ packets. The total forwarding energy $u(k)$ at a non-leaf node k in the super round s is expressed at the lower case in (16).

C.2 Second Step

The second step generates a greedy chain. Two sub-steps are provided for better understanding. We firstly define the constrained Kruskal's MST algorithm that generates an initial chain ignoring the node levels in subsection IV-C.2.a. A complete algorithm that saves energy of depleted level-1 and level-2 nodes is mentioned in subsection IV-C.2.b.

C.2.a **Constrained Kruskal's MST Algorithm.** The constrained Kruskal's MST algorithm generates a chain using the Kruskal's MST algorithm in the graph weighted by $C(*,*)$. The constrained version additionally requires the node degree of every non-leaf node should be 2.

The algorithm collects links that satisfy i) the node degree of the two endpoints of the link is 0 or 1, and ii) the link addition does not generate a loop. And it picks up the shortest link among candidates. The selected link is added to the current working chain under construction one by one until no more selection is possible.

The detailed operation is following. The algorithm uses variables A and p_L. A is a set of nodes whose current node degrees are 0 or 1. A pair of leaf nodes i and j in a chain store their peer

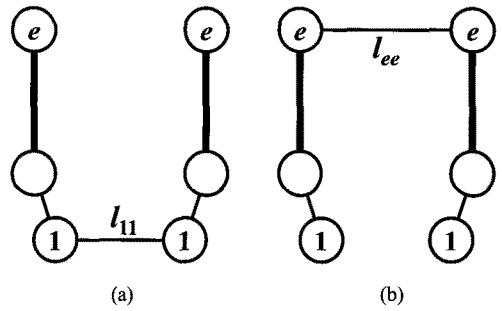


Fig. 3. Leaf node setup: (a) Before, (b) after.

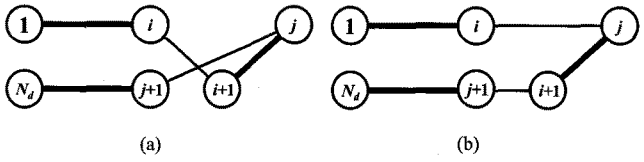


Fig. 4. Link swap: (a) Before, (b) after.

leaf node numbers in $p_L(\cdot)$, i.e. $p_L(i) = j$ and $p_L(j) = i$. The algorithm starts with some preassigned links to be mentioned in subsection IV-C.2.b. The basic operation is to add a link to the current sub-chain. It finds a link $l(i,j)$ among links $l(*,*)$ whose $C(i,j)$ is minimum, $i,j \in A$ and $p_L(i) \neq j$. If found, add $l(i,j)$ to the current sub-chain and update $f(\cdot)$, $p_L(\cdot)$, and A. Checking $p_L(i) \neq j$ guarantees no loop in a chain.

C.2.b **Saving Energy of Depleted Nodes.** This part describes a complete initial chain algorithm that saves forwarding energy at level-1 and level-2 nodes.

Fig. 3 explains how two level-1 nodes are exactly assigned to two leaf nodes. In Fig 3, a bold line indicates a path that consists of a sequence of links and a normal line means a single link. Two level-1 nodes are denoted as '1's and two edge nodes that are randomly generated by the chain algorithm mentioned in subsection IV-C.2.a are represented as 'e's. The algorithm in subsection IV-C.2.a starts with an unconnected initial tree. It includes the link l_{11} in Fig. 3(a) and pre-assigned links chosen by all level-1 and level-2 nodes starting from themselves. The link pre-assignment should keep the node degree constraint and no loop constraint in subsection IV-C.2.a.

After the algorithm in subsection IV-C.2.a generates a complete spanning chain, we substitute l_{ee} in Fig. 3(b) for l_{11} in Fig. 3(a). Regardless whether a level-1 node is a leaf node or not in Fig. 3(a), level-1 nodes are assigned as leaf nodes with this link substitution.

It is important to utilize the Kruskal's MST algorithm instead of the Prim's. These famous algorithms are both greedy and add the shortest link selected in the permitted link pool to the sub-tree under construction. Their difference lies in whether unconnected sub-trees are permitted during operation. Our procedure asks for this property to include unconnected preassigned links as an initial graph. The Kruskal algorithm only allows it.

The constrained Kruskal algorithm has the other good property. By allowing non-connectivity, the constrained Kruskal algorithm is better than the constrained Prim's algorithm in producing short chains. The constrained algorithms are no longer optimum due to the node degree constraint added. Because the constrained Kruskal algorithm has a larger link pool to select a link from, it can choose shorter links and finally produces shorter chains. This property is verified by simulations.

C.3 Third Step

The chain algorithm in subsection IV-C.2 is greedy. All non-optimum greedy algorithms have a bad property to have long links at a few of last choices. The link swap described in Fig. 4 is very effective especially to trim very long links. In Fig. 4, a bold line indicates a multi-link path and a normal line means a single link. In Fig. 4(a), nodes are numbered $1, 2, \cdots, N_d$ along the chain. We use this node position numbers in Fig. 4(b) and in the definition below. The long link $l(j, j+1)$ and a link $l(i, i+1)$ in Fig. 4(a) are replaced, and a shorter chain appears in Fig. 4(b).

The link swap requires three conditions C_1, C_2 and C_3. We assume $C(j, j+1) > C(i, i+1)$ in these definitions.

- $C_1 : C(i, i+1) + C(j, j+1) > C(i, j) + C(i+1, j+1)$
- $C_2 : \max(C(i, i+1), C(j, j+1)) \geq \max(C(i, j), C(i+1, j+1))$
- $C_3 : l(i, i+1)$ or $l(j, j+1)$ must not be a pre-assigned link.

C_1 confines the link swap to reducing the chain length, and C_2 prohibits the appearance of a longer link after the link swap. C_3 guarantees that pre-assigned links survive in the final chain. Note that the link swap requires to update the node position numbers $i+1, i+2, \cdots, j$ in Fig. 4(a) to $j, j-1, \cdots, i+1$ in Fig. 4(b).

Let us define the complete link swap algorithm. The queue Q sorts links $l(*, *+1)$'s with the reversed order of $C(*, *+1)$. Call the node number q_i the ith longest link stored at the ith position in Q. The first pivot link is q_1. If q_j becomes a pivot link, l_q stores $C(q_j, q_j+1)$. The algorithm exhaustively compares a pair of links $\{q_j, q_i\}$, $(j < i)$ with a pivot link q_j.

For a fixed a pivot node q_j, we check whether q_i $(j < i)$ that satisfies C_1, C_2 and C_3 exists. If no such q_i exists, update the pivot link q_j to the next link q_{j+1} in Q. If $\{q_j, q_{i_k}\}$ satisfies C_1, C_2 and C_3, store q_{i_k} and update q_i to q_{i+1}. After testing with every q_i, the link swap is executed only to $\{q_j, q_{i_x}\}$ which most reduces the chain length among every q_{i_k} stored. (17) explains how to choose q_{i_x} among q_{i_k}.

$$C(q_{i_x}, q_{i_x}+1) - C(q_{i_x}, q_j) - C(q_{i_x}+1, q_j+1) = \max_k \left[C(q_{i_k}, q_{i_k}+1) - C(q_{i_k}, q_j) - C(q_{i_k}+1, q_j+1)\right]. \quad (17)$$

After a link swap, Q is resorted and the pivot link is updated to $q_{j'}$ whose $C(q_{j'}, q_{j'}+1)$ is the longest but no longer than l_q. If the shortest link becomes a pivot link, check whether at least a link swap occurs since the longest link was used as a pivot link. If it occurs, use q_1 as a pivot link and repeat. If not, finish the operation.

V. SIMULATION AND ANALYSIS

We use the simulation model as the WSN model described in Section III. Especially, we use i) all nodes are equipped with an identical initial energy E_0 and ii) energy is spent for transceiving only according to (2) and (3). We do not include reconfiguration

energy, because it is hard to define and estimate it, and CREEC uses negligible reconfiguration energy for its lifetime.

The adjacency list at every node is reported to the BS before the first round. The length of adjacency list influences forwarding energy and WSN lifetimes. We can think two kinds of lengths—the conservative length and the practical length. The conservative length is the one that guarantees the least forwarding energy. The practical length uses the reduced list size, which causes larger than the minimum forwarding energy but smaller than the 101% of it. The conservative length and the practical length of adjacency list in a 100-node square WSN are found to be 20 and 10 respectively in CREEC. In a 400-node square WSN, they change to 30 and 20. This simulation only uses the conservative length. Also it shows that the selected link for a chain in CREEC is approximately the 2.7th shortest from the connecting node in 100-node and 400-node WSNs.

We define the WSN condition using 4 parameters—(E_0, D_B, S_d, N_d). They are the initial energy level, the distance from the sensor field to the BS, the side length of the square sensor field, and the number of nodes. Fig. 1 includes the last three parameters. We use 100 independent WSNs for each WSN condition. And node lifetimes and energy consumption patterns are collected as results.

Simulators has been implemented for LEACH, PEGASIS, PEDAP, PEDAP-PA, and CREEC. We collect simulation results from them and provide a table and graphs. Table 1 compares node lifetimes in various WSN conditions. We draw the lifetime distribution in Fig. 5, the consumed energy distribution at 600th round in Fig. 6, and the forwarding energy distribution in Fig. 7.

Table 1 includes four types of node lifetimes. They are the first node death T_1, the $N_d/8$th node death $T_{*/8}$, the $N_d/4$th node death $T_{*/4}$, and the average node death T_{avg}. Table 1 does not use the last node death, because a WSN which contains few nodes is useless, and increasing the last node death can be easily achieved if nodes hibernate properly.

Table 1 describes 6 kinds of WSN conditions. We used (0.25 J, 100 m, 50 m, 100) as a default set of parameters. Other WSN conditions are derived from the default by multiplying or dividing a single parameter by 4. The changed parameter is emphasized with underlining in the first column of Table 1. We compared lifetimes of 5 WSN routing schemes in Table 1. A larger number implies a better result. The best number at each WSN condition in Table 1 is emphasized with bold characters and underlining. We summarize the information obtained from Table 1 as below.

- From the point of T_1 and T_{avg}, we deduce

$$\text{CREEC} > \text{PEGASIS} > \text{LEACH}.$$

- CREEC has the longest T_1 at all WSNs. T_1 in CREEC is 1.3 to 5 times, 1.2 to 6 times and 2 to 49 times longer than T_1 in LEACH, PEGASIS, and PEDAP, respectively.
- A WSN empowered by $4E_0$ has 4 times longer lifetime. The lifetime is almost linear to the initial energy level.
- CREEC has the longest average node lifetime except in a very sparse WSN with a long S_d.

Table 1. Node lifetime for various WSN conditions.

Condition	Routing scheme	Node lifetime			
		T_1	$T_{*/8}$	$T_{*/4}$	T_{avg}
E_0 : 0.25 J D_B : 100 m S_d : 50 m N_d : 100	LEACH	533	568	585	636
	PEGASIS	612	997	1019	1035
	PEDAP	111	798	811	1013
	PEDAP-PA	202	866	1048	1043
	CREEC	**1046**	**1056**	**1061**	**1068**
E_0 : **1.00 J** D_B : 100 m S_d : 50 m N_d : 100	LEACH	2184	2291	2339	2539
	PEGASIS	2450	3994	4080	4146
	PEDAP	443	3224	3301	4169
	PEDAP-PA	3070	3807	4140	4224
	CREEC	**4245**	**4258**	**4265**	**4275**
E_0 : 0.25 J D_B : **25 m** S_d : 50 m N_d : 100	LEACH	815	875	897	948
	PEGASIS	658	1133	1164	1169
	PEDAP	622	819	1008	1173
	PEDAP-PA	824	1151	1181	1166
	CREEC	**1180**	**1188**	**1193**	**1200**
E_0 : 0.25 J D_B : **400 m** S_d : 50 m N_d : 100	LEACH	96	104	112	124
	PEGASIS	324	408	417	433
	PEDAP	8	90	163	273
	PEDAP-PA	8	91	166	278
	CREEC	**367**	**403**	**422**	**439**
E_0 : 0.25 J D_B : 100 m S_d : **200 m** N_d : 100	LEACH	101	143	167	208
	PEGASIS	93	405	517	589
	PEDAP	101	600	699	**775**
	PEDAP-PA	136	**648**	**753**	759
	CREEC	**531**	634	679	718
E_0 : 0.25 J D_B : 100 m S_d : 50 m N_d : **400**	LEACH	544	606	632	694
	PEGASIS	626	1176	1183	1187
	PEDAP	111	814	931	1105
	PEDAP-PA	188	1153	1163	1136
	CREEC	**1182**	**1191**	**1193**	**1198**

Next, we explain why CREEC may not be the best in sparse WSNs. If we denote F_X by the forwarding energy spent for a round in a spanning tree X,

$$F_{\text{MST}} < F_{\text{chain(CREEC)}} < F_{\text{chain(PEGASIS)}}.$$

Forwarding energy in (7) has distance-related and distance-unrelated terms. When S_d is 50 m long, the distance-unrelated term is major. F_{MST}, $F_{\text{chain(CREEC)}}$, and $F_{\text{chain(PEGASIS)}}$ have similar values. They are 20.09 mJ, 20.52 mJ, and 20.87 mJ, respectively. If we increase S_d by 4 times to 200 m, the distance-related term increases by 16 times. $F_{\text{chain(PEGASIS)}}$ and $F_{\text{chain(PEGASIS)}}$ increase to 31.24 mJ and 37.17 mJ which are 128.0% and 152.3% of F_{MST} (= 24.40 mJ), respectively. The increased forwarding energy gap cannot be reduced by CREEC's feedback. In this WSN condition, PEDAP and PEDAP-PA show longer average node lifetimes than CREEC. We are not sure whether PEDAP and PEDAP-PA are definitely better than CREEC, because the long lifetimes in PEDAP and PEDAP-PA are achieved with free frequent reconfigurations. The numbers of reconfigurations for a lifetime are measured as approximately 90, 70, and 10 for PEDAP, PEDAP-PA, and CREEC respectively in the default WSN.

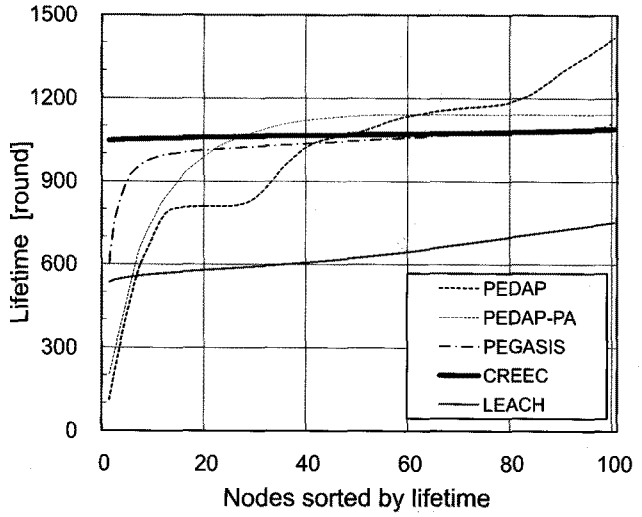


Fig. 5. Node lifetime after all node deaths if (E_0, D_B, S_d, N_d) is (0.25 J, 100 m, 50 m, 100).

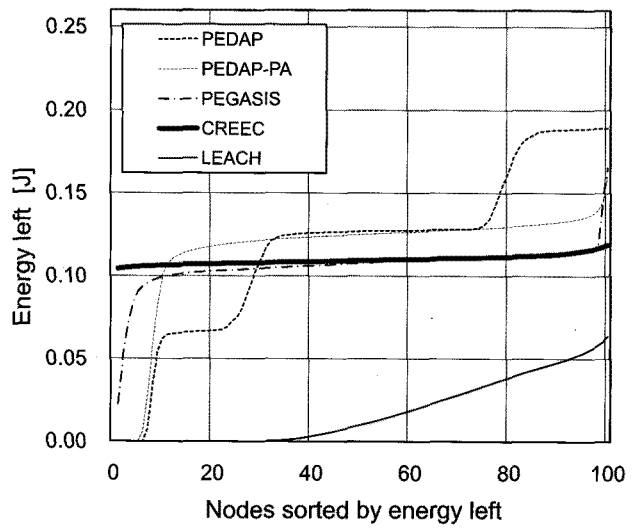


Fig. 6. Distribution of energy left for all nodes at the 600th round if (E_0, D_B, S_d, N_d) is (0.25 J, 100 m, 50 m, 100).

Fig. 5 shows the distribution of node lifetime in the default WSN. (x, y) in Fig. 5 means the xth node death occurs at the yth round. In Fig. 5, LEACH's curve leans to one side and locates at the lower position than CREEC's curve. This shows LEACH has a shorter lifetime with more unfair energy distribution than CREEC.

Fig. 6 shows how much energy is left for all nodes at the 600th round in the default WSN. All nodes are sorted according to increasing order of their remaining energy. Their averages are displayed along the horizontal axis in Fig. 6. CREEC's curve is more horizontal than other curves in Fig. 6. This demonstrates CREEC has very even node lifetimes. CREEC's super round is not exactly 100 rounds long, so the 600th round is not the exact boundary of super rounds. CREEC's curve would be more horizontal if we measure data exactly at the end of the 6th super round.

LEACH's curve starts at 30 nodes in Fig. 6. This means

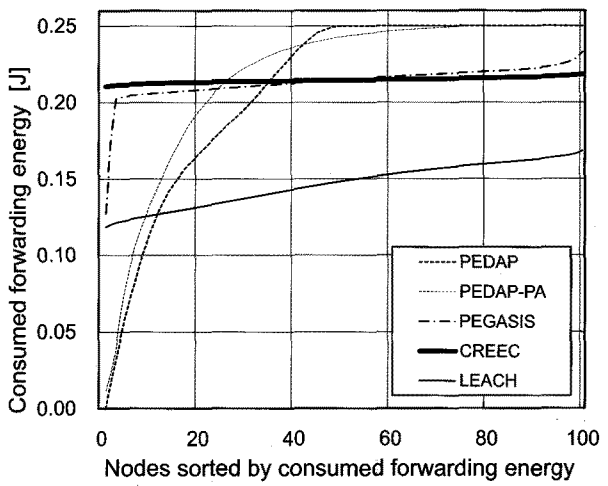


Fig. 7. Forwarding energy distribution after all node deaths if (E_0, D_B, S_d, N_d) is (0.25 J, 100 m, 50 m, 100).

that 30 nodes are already completely depleted before the 600th round. PEDAP's curve and PEDAP-PA's curve look like three-step stairs. In the default WSN, the node degree influences the forwarding energy. Each step from top to bottom corresponds to a node group whose node degrees are 1, 2 and more than 2, respectively. PEDAP-PA's curve has a wider middle step than PEDAP's curve. This means PEDAP-PA is better than PEDAP in fair energy consumption. Because PEDAP-PA's curve still has two other distinct steps, PEDAP-PA cannot be better than CREEC.

Fig. 7 shows forwarding energy measured after all nodes die. CREEC's curve is the most flat among curves in Fig. 7 too. PEGASIS's curve is generally flat and has a cliff pattern at the left side. Two leaf nodes having the least node degree are unchanged for their lifetime. Thus they save much forwarding energy and contribute to generating the cliff pattern. At the right sides of PEDAP's curve and PEDAP-PA's curve, we can check many nodes spend all of their energy 0.25 J for forwarding. 25% of nodes in PEDAP-PA and 50% of nodes in PEDAP do not perform throwing at all.

VI. CONCLUSIONS

We proposed CREEC as a solution for the LEACH-style full-fusion WSN convergecast routing problem. We assume that the base station can predict and simulate transmission energy, so the base station provides a better path to maintain the same energy level at all nodes. CREEC is characterized by fair energy consumption at every node.

To increase WSN lifetime, CREEC proposes the following ideas. CREEC divides a convergecast operation into forwarding and throwing. Every node spends the same throwing energy and the same forwarding energy. CREEC generates chains starting with an initial chain using a constrained Kruskal algorithm and reducing the chain length by a link trimming algorithm called link swap. CREEC updates chains in every super round and builds new chains to save energy at depleted nodes. Equipped with these techniques, CREEC shows longer lifetimes than other

schemes with increased convergecast delay.

Simulations confirm our assertion. CREEC has longer average lifetime and longer first node death than LEACH, PEGASIS, PEDAP, and PEDAP-PA under various WSN conditions with various initial energy, BS locations, side lengths of square WSN fields, and the number of nodes, except in a few sparse sensor networks. We verified that CREEC is very excellent for convergecasts in non-time critical homogeneous sensor networks.

REFERENCES

[1] A. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[2] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," in *Proc. Int. Workshop Wireless Sensor Netw. Appl.*, Sept. 2002, pp. 22–30.

[3] J. Kulik, W. Heinzelman, and H. Balakrishman, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Netw.*, vol. 8, no. 2/3, pp. 169–185, Mar. 2002.

[4] R. Vidhyapriya and P. T. Vanathi, "Conserving energy in wireless sensor networks," *IEEE Potentials*, vol. 26, no. 5, pp. 37–42, Sept. 2007.

[5] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A Survey," *IEEE Wireless Commun.*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[6] Z. Cheng, M. Perillo, and W. B. Heinzelman, "General Network lifetime and cost models for evaluating sensor network deployment strategies," *IEEE Trans. Mobile Comput.*, vol. 7, no. 4, pp. 484–497, Apr. 2008.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient communication protocol for wireless micro sensor networks," in *Proc. the 33rd Ann. Hawaii Int. Conf.*, Jan. 2000, pp. 3005–3014.

[8] H. O. Tan and Ibrahim Korpeoglu, "Power efficient data gathering and aggregation in wireless sensor networks," *ACM SIGMOD Record*, vol. 32, no. 4, pp. 66–71, Dec. 2003.

[9] S. Lindsey, C. Raghavendra, and K. Sivalingam, "Data gathering algorithms in sensor networks using energy metric," *IEEE Trans. Parallel and Distributed Syst.*, vol. 13, no. 9, pp. 924–935, Sept. 2002.

[10] S. Upadhyayula and S. K. S. Gupta, "Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (DAC) in wireless sensor networks," *Ad Hoc Netw.*, vol. 5, no. 5, pp. 626–648, July 2007.

[11] N. Tabassum, Q. E. K. M. Mamun, and Y. Urano, "COSEN: A chain oriented sensor network for efficient data collection," in *Proc. ITNG*, Apr. 2006, pp. 262–267.

[12] L. Yuan, Y. Zhu, and T. Xu, "A multi-layered energy-efficient and delay-reduced chain-based data gathering protocol for wireless sensor network," in *Proc. MESA* , Oct. 2008, pp. 13–18.

[13] W. B. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

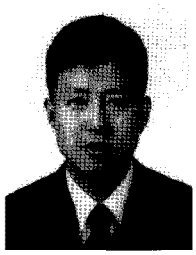
[14] C.-K. Liang, Y.-J. Huang, and J.-D. Lin, "An energy efficient routing scheme in wireless sensor networks," in *Proc. Fifth Int. Conf. Comput. Sci. Appl.*, Mar. 2008, pp. 916–921.

[15] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Technical J.*, vol. 36, pp. 1389–1401, 1957.

[16] L. H. A. Correia, D. F. Macedo, A. L. dos Santos, A. A. F. Loureiro, and J. M. S. Nogueira, "Transmission power control techniques for wireless sensor networks," *Computer Networks: The Int. J. Comput. Telecommun. Netw. Archive*, vol. 51, no. 17, pp. 4765–4779, Dec. 2007.

[17] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," in *Proc. the American Math. Soc.*, vol. 7, Feb. 1956, pp. 48–50.

Jisoo Shin received B.Sc., M.Sc., and Ph.D. in Department of Computing from Soongsil University of Seoul, Korea, in 2003, 2005, and 2009, respectively. He worked at Gensolsoft corporation, Korea, from 2005 to 2010. He has been a Researcher of Soongsil University and a Senior Member of ETRI, Korea, from 2009 and 2010, respectively. His major research interests include sensor networks, future internet, carrier ethernet, and routing theory.



Changjin Suh received his B.Sc. and M.Sc. from Seoul National University, Korea in 1982 and 1984 respectively. He received Ph.D. in ECE (Electric and Computer Engineering) from the University of Massachusetts in Amherst, U.S.A. in 1996. From 1985 to 1990, he worked at ETRI (Electronics and Telecommunications Research Institute), Korea. In 1997, Dr. Suh joined the Department of Computing, Soongsil University, Korea as a Professor. His research interests focus on sensor networks, carrier ethernet, switching, routing theory, and BGP.