

An Adaptively Balancing Workload Protocol on Query Trees for Maximizing Lifetime in Sensor Networks

Anuparp Boonsongsrikul

Dept. of Electrical Engineering,
Burapha University, Thailand
Email: anuparp@eng.buu.ac.th

Seung-Kyu Park

Graduate School of Information and
Communication, Ajou University, Korea
Email: sparky@ajou.ac.kr

Seung-Hun Shin

Div. Information and Computer Engineering,
Ajou University, Korea
Email: sihnsh@ajou.ac.kr

Abstract—Since querying trees in wireless sensor networks are normally built in an ad-hoc manner, equal workload among sensors cannot be guaranteed. A sensor with more descendants has to relay more data that cause its energy exhausted more quickly. To prolong the lifetime of such a sensor and its network we propose an adaptive balancing workload protocol consisting of two phases. First, it builds a balanced-query tree that allows nodes close to a sink to share their workload. Second, it adaptively switches the workload from an exhausted node to non-exhausted nodes. The simulation was carried out to validate the proposed protocol and compare it with previous work. The simulation shows that after running 2,100 rounds, the node by our proposed scheme still remains 20% of initial energy while the one by previous schemes runs out of energy fully exhausted.

Index Terms—sensor networks, balancing workloads, energy, network lifetime, query trees

I. INTRODUCTION

Wireless sensor networks are used in a variety of applications that include remote habitat monitoring, battlefield monitoring and environmental data collection (temperature, humidity, light, vibration, etc.) as reported by Akyildiz [1] and Szwedczyk [2]. This paper focuses on the applications of data acquisition systems where data from every node needs to be collected at a sink node. (This excludes the option of data aggregation [3], [4], [5] and [6]).

Query trees are usually built in an ad-hoc manner and therefore there is no guarantee that all sensors can have the equal workload. Recently, Andreou [7] and Chatzimilioudis [8] presented query routing tree construction by balancing node degrees. Without balancing node degrees, an internal node may have arbitrary children that can suffer from node congestion. It reduces the life time of the network due to the numerous retransmissions from child nodes. The contemporary solutions to balance the node degrees are energy-driven and based on minimum hot-spots.

Andreou and et al. [7] proposed Energy-driven Tree Construction (ETC) consisting of two steps in building the query routing tree. First is to construct an arbitrary tree and; second is to balance node degrees of internal nodes. To balance node degrees, the sink uses the maximum depth of the tree d , and the total number of nodes n , to compute node degree,

$\beta = \sqrt[d]{n}$. For example, if $n = 80$ and $d = 8$, then $\beta = \sqrt[8]{80} = 1.729$. Therefore for an integer number, the node degree can be approximately 2.

Chatzimilioudis [8] proposed an algorithm to build a balanced tree with Minimum Hot-Spots (MHS). The order in which the nodes pick their parent importantly affects the quality of the outcome of minimizing hot spots. Nodes with many incoming streams are primary candidates for network hot-spots. The proposed tree outperforms competition and achieves tree balance. The majority of internal nodes have the degree equal to one.

However, both existing schemes [7] and [8] for constructing query trees do not have a mechanism to prolong a sensors life when a node runs nearly out of energy. Particularly, nodes that are close to the sink do not balance (and/or switch) their workload. This motivates us to propose a novel protocol for adaptively balancing workload on query trees for maximizing the network lifetime in sensor networks.

This paper presents an adaptive balancing workload protocol to prolong the lifetime of the network. The proposed protocol consists of two phases. First, a balanced query tree is built where two selected children by the sink share their workloads and all internal nodes have a limited number of child nodes. Second, the workload is adaptively switched to non-exhausted nodes from an exhausted node. This paper adopts the definition of the network lifetime given by Buragohain [9], Younis [10] and Hou [11]. That is the time duration before the first sensor node runs out of energy.

The rest of this paper is organized as follows. Section II presents the adaptive balancing workload protocol. Section III evaluates the effectiveness of our protocol and shows the simulation results. Section IV concludes the paper.

II. PROPOSED PROTOCOL

This section presents the adaptively balancing workload protocol consisting of two phases (refer to Fig. 1).

- 1) Construction of balanced query tree
- 2) Reconstruction of the tree during service time

In phase 1, a balanced query tree is constructed such that two children of the sink make balance. The balance of

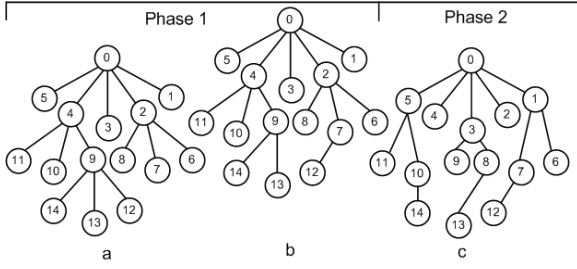


Fig. 1. A model of adaptive balancing workloads

the workloads can be achieved by limiting the number of children in internal nodes. In phase 2, sensor nodes start applications. When one node is notified as near exhausted, the tree is dynamically reconstructed transferring its descendants to another branch

A. Construction of balanced query tree

The balanced query tree is constructed based on breadth-first search. The proposed balanced query tree is similar to the shortest-path spanning tree by Zou [12] except following two differences.

- 1) The degree of nodes is determined during the process of breadth-first search.
- 2) The query tree is divided into two subtrees such that the number of nodes are nearly balanced.

As a result, an internal node limits the number of children, which two selected children by the sink can share their workloads. The tree constructed in the phase 1 becomes the base with which the service of sensor network starts. In phase 1, two algorithms are proposed: 1) Breadth-first search for balancing node degrees and 2) Balancing workloads between two subtrees.

1) *Breadth-first search for balancing node degrees:* The algorithm begins at the sink. The sink invites its neighboring nodes to become its children. A node overhearing invitation sends a childship message including the number of its neighboring nodes to the sink. Let a node know the number of its neighboring node. The sink takes all replying messages and enrolls corresponding nodes to be its children. Among children, it selects two nodes who have the first and second maximum number of neighboring nodes and call them Subtree Heads (SH). Siblings of SHs select themselves to be external nodes. Each SH invites its neighboring nodes to be its children but selects only m nodes where m is a given threshold. Since near-uniform node deployment is one of the easiest and most practical approaches to provide full sensing coverage and connectivity, we will assume that the proposed scheme in this paper applies to a random uniform node deployment and an average number of neighboring nodes is an integer z where $z > 2m$. The algorithm 1 is described in Table I. Fig.1(a) shows a constructed tree after applying algorithm 1, where node 0 is a sink, node 2 and 4 are the SHs, and the node degree is three.

Algorithm 1

- S : The sink
 N_i : Neighboring node(s) of a node i ;
- I. First stage** (1^{st} depth of the tree)
- 1) S invites N_S to be its children by broadcasting an invitation message.
 - 2) N_S replies a childship message to be a child of S .
 - 3) S takes all replying messages and enrolls N_S to be its children. Among N_S , S selects two nodes who have the first and second max. number of neighboring nodes to be Subtree Heads (SH).
 - 4) Siblings of two SHs select themselves to be external nodes
- II. Second state** (2^{nd} depth of the tree);
- 1) SH invites neighboring nodes of SH, NSH , to be its children by broadcasting an invitation message.
 - 2) N_{SH} having no parent and receiving an invitation message replies a childship message to SH .
 - 3) If SH receives the number of the childship messages more than m , SH chooses only m children where m is a given threshold.
- III. Third stage** (q^{th} depth of the tree where $q > 2$);
- 1) Node i having parent invites N_i to be its children by broadcasting an invitation message.
 - 2) N_i having no parent and receiving an invitation message replies a childship message to node i .
 - 3) If node i receives the number of the childship messages more than m , it chooses only m children otherwise it becomes an external node.
 - 4) To reduce orphans (nodes who are not selected by parents).
 - a) Each orphan broadcasts an adopt-me message.
 - b) Each external node that received an adopt-me message replies by sending an adoption message to the orphan.
 - c) The orphan chooses its parent from the adaptation messages and acknowledges the parent.

TABLE I
BREADTH-FIRST SEARCH FOR BALANCING NODE DEGREES

2) *Balancing workloads between two subtrees:* The query tree is divided into two subtrees. A subtree is a tree T_L consisting of a SH and all descendants of the SH in T_L where L is a label representing the size of a subtree. To balance workloads of two SHs, there are two steps.

Step 1. Counting the number of nodes in a T_L :

The process of step 1 starts from the leaf nodes towards the sink. Suppose a node has m children and a child i has d_i descendants. A node sends value V to its parent as the following.

$$V = \sum_{i=1}^m (d_i + 1) + 1, \quad \text{for } m \geq 1$$

$$= 1, \quad \text{for } m = 0$$

By hop-by-hop communications from the leaf nodes towards the sink, a SH will finally know the total number of nodes in its subtree. Let K and $mean$ denote the number of nodes in a subtree and an average number of nodes between two subtrees, respectively.

Step 2. Balancing the number of nodes between two subtrees:

Algorithm 2

- 1) Each node
 - a) Counts descendants (if any) and sends the number of descendants plus one to its parent.
 - b) Computes a reference value r after receiving a balancing message $\langle id, mean, K \rangle$ originated by the SH.
 - 2) The donor
 - a) Selects a closest beggar and sends a donation message to him
 - b) Sends a confirmation message to a beggar
 - 3) A beggar receiving a donation message
 - a) Replies to the donor
 - 4) Both donor and beggar report to the sink
-

TABLE II
BALANCING THE NUMBER OF NODES BETWEEN SUBTREES

Two SHs exchange their value K via the sink and compute a mean. If $mean = K$, step 2 stops because two subtrees are balanced; otherwise the SH sends a balancing message to all of its descendants. The format of the balancing message is $\langle pid, mean, K \rangle$ where pid is parent ID . If $mean < K$, a node knows that he stays in the bigger subtree T_b ; otherwise he stays in the smaller subtree T_s .

To balance the number of nodes between two subtrees, the process requires to emigrate nodes from T_b to T_s . Let r be an absolute value of $\lfloor mean - K \rfloor$ representing the balancing number of nodes. Each node computes value r and stores it as a reference.

Definition 1: A candidate donor is an internal node in T_b , who has the number of descendants more than or equal to r . A beggar is a leaf node in T_s , who is in the transmission range of a donor's child.

To balance the number of nodes between two subtrees, four steps are required as follows 1) the donor broadcasts a donation message; 2) a beggar replies to the donor; 3) the donor selects a beggar and sends a confirmation message to a beggar; 4) both the donor and beggar send a balancing report towards the sink via their internal nodes. Since there are multiple candidates for donors and beggars, the selection of a donor and a beggar can be explained as follows.

1. Selecting a donor.

A candidate donor sends the number of descendants to the sink. The sink selects a donor based on the priority as follows. First, a candidate donor who has the number of descendants equal to r . Second, if there is no candidate donor in the first priority, the sink selects a donor who provides the number of descendants closest to r . Third, if there are two or more candidate donors providing the same number of descendants, the sink randomly selects one of them.

2. Selecting a beggar.

The donor selects a beggar which is closest to him. A selected beggar must be within the transmission range of all the donated children. If the donor cannot find such a beggar then he alternatively chooses other few beggars and distributes the donated children to them. The format of a donation message is $\langle id, D, dd \rangle$ where id is node ID , D

Algorithm 3

- When remaining energy is lower than the threshold,
- 1) An exhausted node sends a help-me message to its parent
 - 2) A node sends overhearing a help-me message sends a willing message $\langle id, energy, Q \rangle$ to the parent of an exhausted node
 - 3) The parent distributes workloads of an exhausted node to nodes based on remaining energy and Q
-

TABLE III
DYNAMICALLY BALANCING WORKLOADS

is a set of donated children and dd is the number of donated descendants. Let N be a set of neighboring nodes of a beggar and W be a set of $D \cap N$. After receiving a donation message, a beggar sends W to the donor. The donor selects a beggar who gives $W = D$.

In the end of this stage, the donor sends a confirmation message to the beggar. Then both the donor and beggar send two balancing reports to the sink. Balancing the number of nodes between two subtrees can be shown in Algorithm 2 (Table II).

Fig.1(b) shows the migrated tree after applying algorithm 2 to the Fig.1(a). The number of nodes in T_s and T_b are 4 and 7, respectively. Since mean is 5.5, so the required number of nodes transferring from T_b to T_s is 1. Node 9 is the donor. Node 7 is the beggar. Node 9 gives its child (node 12) to node 7.

B. Reconstruction of the tree during service time

Phase 2 is activated while a normal service of sensor network runs applications. The sensing messages from nodes are collected to a sink node. Since the nodes located near to the sink have more message traffics, they are prone to more energy consumption. When a node is battery exhausted, it requires a mechanism to prolong the lifetime of a sensor. This can be possible by unloading its workload to non-exhausted nodes. Let a set of workloads W denote a list of children of an exhausted node and N denote a list of neighboring nodes. The unloading process starts from nodes close to the sink because two SHs have the heaviest burdens. When energy of a node is lower than a given threshold, a node sends a help-me message $\langle id, energy, W \rangle$ to its parent where id is node ID and $energy$ is remaining energy which is lower than the threshold. If a node overhears that a workload is in its transmission range, he sends a willing message $\langle id, energy, Q \rangle$ to a parent of an exhausted node, where energy is remaining energy which is not less than the threshold and Q is overlap between a set of its neighboring nodes and a set of workloads ($N \cap W$). If there is a member(s) in Q , it represents that such a node and child of an exhausted node are in the transmission range. Since a parent knows nodes in terms of remaining energy and Q , he can distribute workloads from an exhausted node (its child) to non-exhausted nodes.

Fig. 1(c) shows the dynamically balanced tree after applying algorithm 3 to the Fig. 1(b). When node 2 and 4 are exhausted,

they send a help-me message to node 0. Workloads of node 2 are switched to node 1 and 3 and workloads of node 4 are switched to node 3 and 5. Similarly, when node 9 is exhausted, a workload of node 9 is moved to node 8.

III. SIMULATION AND EVALUATION

To simulate the lifetime in the dynamically balanced trees, the sensor network is modelled. The simulation was carried on to validate the query tree construction, measure the degree of load balance using the balance index [13] and evaluate the network lifetime using ns-2 [14]. The previously proposed methods by related researches are also simulated to compare the performances.

A. Validation query tree construction

Simulating with different node degrees in the balanced query tree, similar performance trends were observed. Unbalanced numbers of nodes between two subtrees are improved after applying algorithm 2. Table IV compares properties of the balanced query tree generated by algorithm 1 and 2 where K_{Tb} / K_{Ts} denotes a ratio of the number of nodes in a subtree T_b to the number of nodes in a subtree T_s while Ψ denotes a ratio of the number of leaf nodes to the number of parent nodes. In setting up topologies, the transmission range of a node is fixed at 10 m and the number of nodes is 80, an area is $40 \times 40 m^2$ while node degrees are varied.

We observe that if $\Psi \geq 2$ representing the number of leaf nodes is at least twice as large as the number of parent nodes, leaf nodes can be covered a network when some node is exhausted. Scattering leaf nodes over a network helps an exhausted node to find leaf nodes (non-exhausted nodes) easily. For node degree 3, 4 and 5, the topology provides $\Psi > 2$. For simplicity, let us select node degree (m) = 3 to present an example of our balanced tree.

Fig. 2, 3 and 7 show an example of our balanced tree. A sink is located at coordinate (0,0) and other nodes are randomly deployed where m = 3. Dashed edges represent a subtree which a donor (dark) stays while solid edges represent a subtree which a beggar (gray) stays.

A constructed tree initially consists of $K_{Tb} = 49$ and $K_{Ts} = 27$ as illustrated in Fig. 2. Since K_{Tb} / K_{Ts} , a dark node gives 11 descendants to a gray node. Then, $K_{Tb} = K_{Ts} = 38$ that the topology is illustrated in Fig. 3. Note that there is only a child left at the dark node while there are 11 descendants moved to the gray node. Besides, a node has children less than three, its child was an orphan.

B. Measurement of the balance index

This section measures the degree of load balance of a sensor network by adopting the *load balance*, $\beta(E)$, from [13]. The balance index is a fairness index among loads in subtrees:

$$\beta(E) = \frac{(\sum K_{SH_i})^2}{x \sum K_{SH_i}^2}$$

where SH_i is Subtree Head of (top) subtree i , K_{SH_i} is the total number of workloads of SH_i , and x is the number

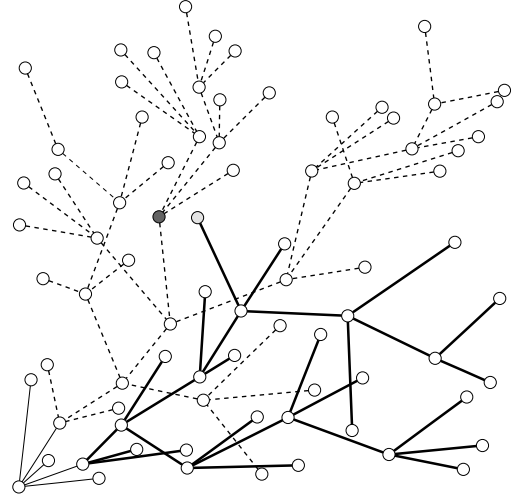


Fig. 2. A query tree by Algorithm 1

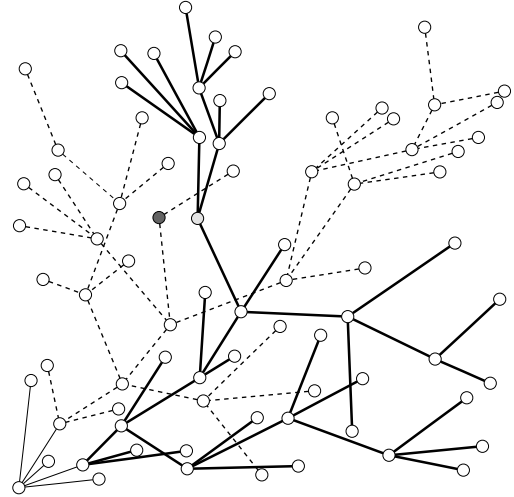


Fig. 3. A query tree by Algorithm 2

of subtrees. The balance index has the desired value that it approaches to 1 when workloads of subtrees are nearly balance. This implies that the less difference between K_a and K_b , the more improvement in load balancing. In this evaluation of the balance index, 80 - 200 sensor nodes are randomly deployed in a square area. After constructing the query trees based on protocols of [7], [8] and our work, The balance index is used to evaluate the performance for [7], [8] and our work.

Fig. 4 shows the balance index where the sink is located in the center of an area. The balance index of our work approaches to 1 while that of [7] and [8] is 0.77 and 0.74 on average, respectively. The main reason that the balance index of our work is slightly better balanced on average than both [7] and [8] because our algorithm 1 offers load balancing at the top subtrees while both [7] and [8] do not provide load balancing on top of subtrees.

Fig. 5 shows the balance index where the sink is located in the corner of an area. The balance index of our work is also

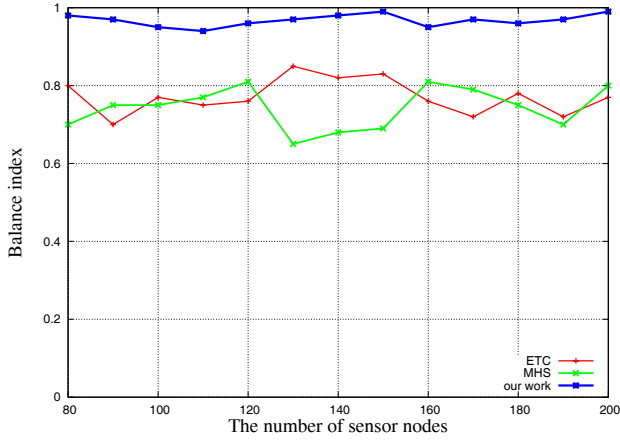


Fig. 4. Balance index where the sink is located at the center

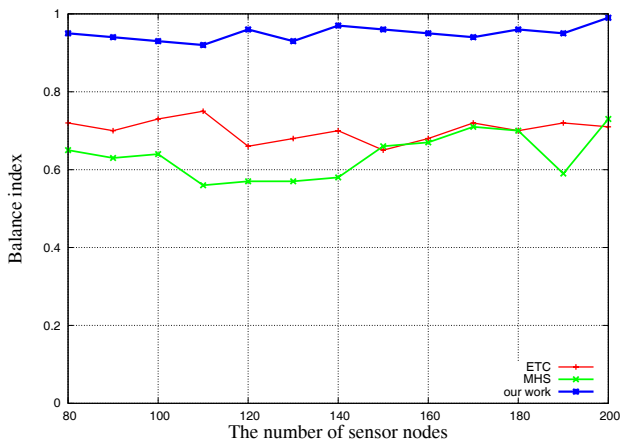


Fig. 5. Balance index where the sink is located at the corner

better balanced on average than both [7] and [8].

C. Network lifetime

This section uses energy models of Heinzelman [15]. The energy in sending a message of a node is $s \cdot (\delta + \theta \cdot d^q)$, where s is the message size, $\delta = 50$ nJ/b is a distance-independent

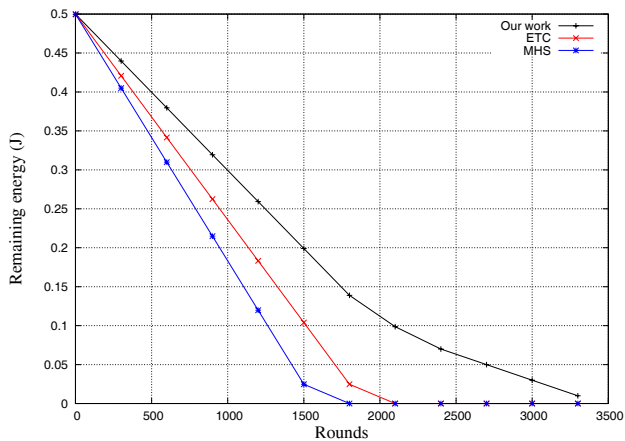


Fig. 6. Network lifetime

TABLE IV
VALIDATION OF ALGORITHM 1 AND 2

Node degree m	Algorithm 1		Algorithm 2	
	K_{Tb} / K_{Ts}	Ψ	K_{Tb} / K_{Ts}	Ψ
2	1.69	1.13	1.12	1.13
3	1.80	2.16	1.00	2.03
4	1.43	3.04	1.09	3.15
5	1.56	3.67	1.04	3.54

term, $\theta = 100$ pJ/b/m is the coefficient for a distance-dependent term, $q = 2$ is the exponent for the distance-dependent term, and d is the transmission distance. The energy in receiving a message of a node is $s \cdot \gamma$, where $\gamma = 50$ nJ/b is a coefficient independent of transmission distance. In the experiments, the default message size is set at 48 bytes Madden [16]. The initial energy budget at each sensor node is set at 0.5 J. A node switches its workloads when remaining energy is 0.1 J. The examples of balanced trees ETC [7] and MHS [8] are illustrated in Fig 8 and 9, respectively. Since nodes close to the sink do not balance their workloads, a black node has the heaviest workloads and results in its lifetime being shortest.

By comparing with ETC and MHS, our scheme balances workloads of nodes close to the sink and distributes workloads of an exhausted node. Our balance query tree is gradually changed from Fig. 3 to 7. This makes an exhausted node have a longer life and result in the longer lifetime of a network. Fig. 6 compares remaining energy of a node having the heaviest workloads in ETC, MHS and our scheme. We run 3,500 rounds in observing remaining energy. In a round, each node has to send its own reading message to the sink. Note that at the 2,100th round a black node in ETC and MHS already run out of energy while a node in our scheme is still alive.

IV. CONCLUSION

Unbalanced workload is an important problem in wireless sensor networks, which can dramatically reduce the network lifetime. In this paper, we present an adaptive balancing workload protocol on a query tree in wireless sensor networks.

We address the insufficiencies of recently previous work and propose an algorithm to balance workloads of nodes close to the sink and to distribute workloads from an exhausted node to non-exhausted nodes. By comparing with related work, our resulting balanced tree achieves to prolong the network lifetime. The simulation shows that after running 2,100 rounds, the node by our proposed scheme still remains 20% of initial energy while the one by previous schemes runs out of energy fully exhausted.

REFERENCES

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, vol. 40, pp. 102–114, 2002.
- [2] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, J. S. Jacinto, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Magazine Communications of the ACM - Wireless sensor networks*, vol. 47, pp. 34–40, 2004.

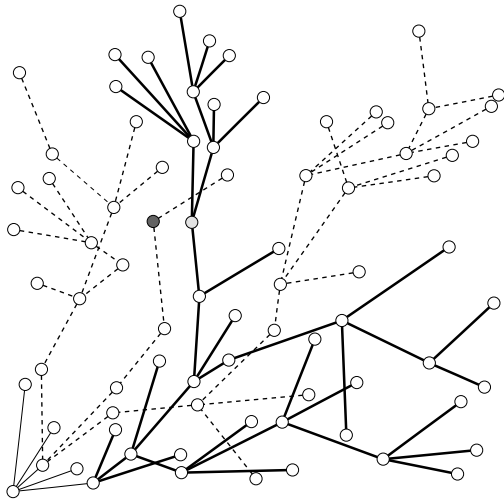


Fig. 7. A query tree by Algorithm 3

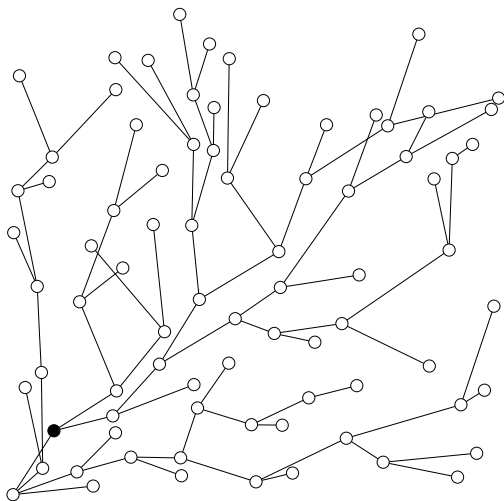


Fig. 8. A balance tree of ETC

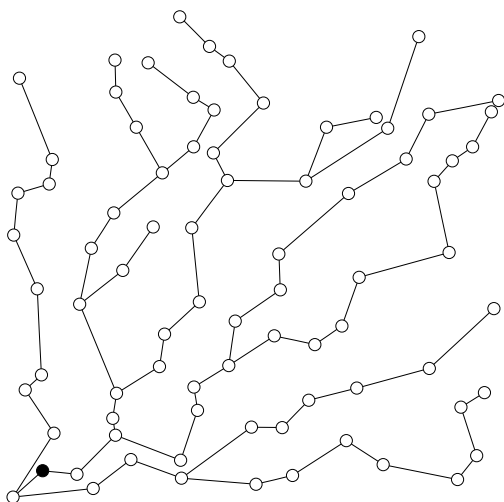


Fig. 9. A balance tree of MHS

- [3] A. Boonsongsrikul, K. suk Lhee, and S.-K. Park, "Monitoring-Based Secure Data Aggregation Protocol against a Compromised Aggregator in Wireless Sensor Networks," *Journal of Korea Information Processing Society*, vol. 18-c, pp. 303-316, Oct. 2011.
- [4] A. Boonsongsrikul, K. suk Lhee, and M. Hong, "Securing Data Aggregation against False Data Injection in Wireless Sensor Networks," in *Proceedings of the 12th International Conference on Advanced Communication Technology (ICACT2010)*, 2010, pp. 29-34.
- [5] A. Boonsongsrikul, S.-K. Park, and S.-H. Shin, "An Approach of False Data Identification Protocol for Minimum Communication Cost in Wireless Sensor Network," *Journal of Korea Society of Computer and Information*, vol. 10, pp. 121-129, Oct. 2011.
- [6] A. Boonsongsrikul, S.-K. Park, and S.-H. Shin, "A Sextant Cluster Based Monitoring on Secure Data Aggregation and Filtering False Data in Wireless Sensor Networks," *Journal of Korea Society of Computer and Information*, February 2012, to be published.
- [7] P. Andreou, A. Pamboris, D. Zeinalipour, P. Chrysanthis, and G. Samaras, "ETC: Energy-Driven Tree Construction in Wireless Sensor Networks," in *Proceedings of Mobile Data Management: Systems, Services and Middleware (MDM '09)*, 2009, pp. 513-518.
- [8] G. Chatzimilioudis, D. Zeinalipour-Yazt, and D. Gunopulos, "Minimum-Hot-Spot Query Trees for Wireless Sensor Networks," in *Proceedings of the 9th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'10)*, 2010.
- [9] C. Buragohain, D. Agrawal, and S. Suri, "Power aware routing for sensor databases," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'05)*, 2005, pp. 1747-1757.
- [10] Y. O. and F. S., "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'04)*, 2005, pp. 629-640.
- [11] Y. T. Hou, Y. Shi, and H. D. Sherali, "Rate allocation in wireless sensor networks with network lifetime requirement," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing, (MobiHoc'04)*, 2004, pp. 67-77.
- [12] S. Zou, I. Nikolaidis, and J. J. Harms, "ENCAST: EnergyCritical Node Aware Spanning Tree for Sensor Networks," in *Proceedings of the CNSR'05*, 2005, pp. 249-254.
- [13] P.-H. Hsiao, A. Hwang, H. Kung, and D. Vlah, "Load-Balancing Routing for Wireless Access Networks," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'01)*, 2001, pp. 986-995.
- [14] <http://www.isi.edu/nsnam/ns/> The Network Simulator-ns-2 [Online].
- [15] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Hawaii Int. Conf. on System Sciences*, 2000, pp. 3005-3014.
- [16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the ACM SIGMOD'03*, 2003, pp. 491-502.