Available online at www.sciencedirect.com
ScienceDirect
Signal Processing 87 (2007) 2934–2948
SIGNAL PROCESSING
www.elsevier.com/locate/sigpro

Minimum-delay energy-efficient source to multisink routing in wireless sensor networks

Shun-Yu Chuang, Chien Chen*, Chang-Jie Jiang

Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan

Received 1 September 2006; received in revised form 19 March 2007; accepted 15 May 2007
Available online 24 May 2007

Abstract

This paper proposes a simple and scalable approach to multisink routing scheme in wireless sensor networks. Wireless sensor network is a rapidly growing discipline, with new technologies emerging and new applications under development. In addition to providing light and temperature measurements, wireless sensor nodes have applications such as security surveillance, environmental monitoring, and wildlife watching. One potential problem in a sensor network is how to transmit packets efficiently from single-source to multi-sinks, i.e., to gather data from a single sensor node and deliver it to multiple clients who are interested in the data. The difficulty of such a scenario is finding the minimum-cost multiple transmission paths. Many routing algorithms have been proposed to solve this problem. Most current algorithms address the reduction of power consumption, and potentially introduce a large delay. This paper proposes a novel multi-path routing algorithm, called hop count based routing (HCR) algorithm, which considers energy cost and transmission delay simultaneously. A hop count vector (HCV) is introduced to support routing decision. Moreover, an additional pruning vector (PV) can further enhance routing performance. The proposed algorithm also provides a maintenance mechanism to handle the consequence of faulty nodes. A failure of a node leads to an inaccurate HCV. Therefore, an efficient correction algorithm is necessary. An Aid-TREE (A-TREE) is applied to facilitate restricted flooding. This correction mechanism is more efficient than full-scale flooding for correcting the limited inaccurate HCVs. Finally, the impact of failed nodes is studied, and an algorithm, called Lazy-Grouping, is proposed to enhance the robustness of HCR.

Keywords: Source to multisink routing; Path aggregation; Hop count vector; Prune vector; Set cover; Grouping; Wireless sensor networks

1. Introduction

Wireless sensor network is a rapidly growing discipline, with new technologies emerging and new applications under development. In addition to providing light and temperature measurements, wireless sensor nodes have applications such as security surveillance, environmental monitoring, and wildlife watching [1–5]. According to the number of sources and sinks in various applications, we can divide the routing in wireless sensor networks into four schemes: (1) single-source to single-sink (SSSS) [6]—the simplest model in routing complexity, (2) multi-source to single-sink (MSSS)—many researchers have provided algorithms

*Corresponding author.
E-mail addresses: is89060@cis.nctu.edu.tw (S.-Y. Chuang), cchen@cis.nctu.edu.tw (C. Chen), y.cis93g@cis.nctu.edu.tw (C.-J. Jiang).

0165-1684/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2007.05.011

to handle this model [7–11], (3) single-source to multi-sinks (SSMS) [12], and (4) multi-source to multi-sinks (MSMS)—the most complicated routing model. Many researchers have attempted to solve the MSMS problem with the help of MSSS or SSMS [7,12,13]. This paper focuses on the potential application model for a sensor network, which is transmitting packets efficiently from SSMS [12], i.e., to gather data from a single sensor node and deliver it to multiple clients who are interested in the data, as illustrated in Fig. 1. An example of such SSMS routing is multiple fire stations (i.e., multi-sinks) that use the sensor network for monitoring fire (i.e., single source) in some regions.

The difficulty of handling this model is in arranging the minimum-delay transmission paths to form a forwarding tree. The simple conventional approach is the brute force (BF) method. As shown in Fig. 2, the source finds individual paths to each sink. This approach chooses many relaying nodes (nine nodes in this example), but has a minimum transmission delay. Fig. 3 shows a power-optimal solution by combining multiple transmission paths in Fig. 2, which has only six relay nodes. Although the energy cost of the power-optimal approach is lower than the BF method in Fig. 2, it has a larger transmission delay. Minimizing the transmission delay is important when a sensor node needs to transmit an urgent message to the sinks. However, saving power is equally important, since most sensor nodes are powered by limited disposable batteries. Hence, the best policy is to consider both energy and delay simultaneously, as illustrated in Fig. 4. The solution in Fig. 4 has only six relay nodes and transmission delay is as short as in the BF method. However, determining when to combine and divide packet transmission paths is a challenge. Attempts to derive an algorithm to optimize the energy consumption and the transmission delay simultaneously have so far proven futile [14]. This paper presents an approach to simultaneously address energy-cost and end-to-end delay.

In this paper, we assume that the transmission delay is proportional to the number of hops. In our wireless sensor networks, we assume that traffic is not heavy, which is a reasonable assumption for the transmission properties of sensor nodes. Hence, the influence of contention and collision on the delay is negligible. Since we assume that each sensor node has the same transmission range, the energy cost is

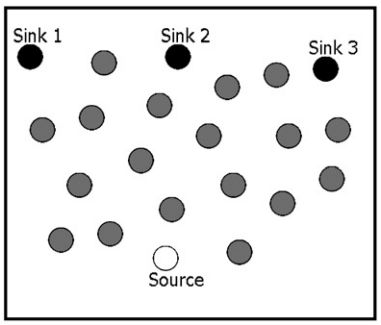

Fig. 1. Example of SSMS model.

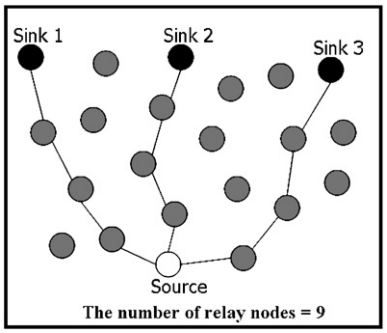

Fig. 2. Transmission paths of BF method.

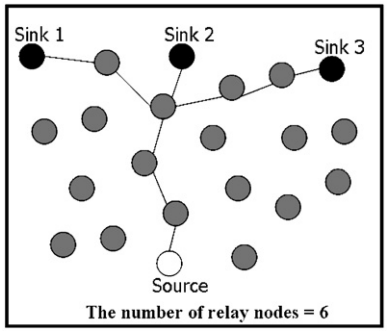

Fig. 3. Transmission paths of power-optimization method.

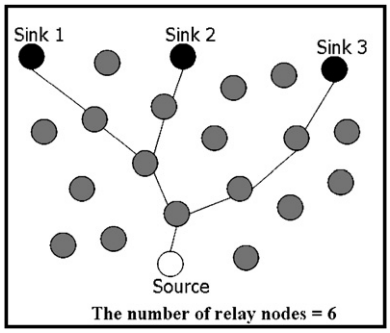

Fig. 4. Transmission paths of the best solution.

defined as the total number of relaying nodes needed to forward the data to all sinks. Therefore, our goal for minimum delay is to find the path with the minimum number of hops traveling from the source to each sink. At the same time, we like to achieve better energy efficiency by involving less relaying nodes in the SSMS routing.

SEAD [12] is a common representative solution to deal with the above SSMS problems. SEAD maintains a D-TREE (Dissemination Tree) for communication. However, SEAD has some drawbacks. The first drawback of SEAD is its large overhead. SEAD constructs a D-TREE, which is a minimum-cost weighted Steiner tree, for an instant source, which transfers packets to multiple sinks, and then constructs another D-TREE if another source is available. Many sensors are available to send information to sinks, making the tree construction overhead excessively large. The second drawback is the large delay. A D-TREE might lower energy consumption, but it also increases end-to-end transmission delay. The third drawback is that SEAD assumes that each sensor node must know its own geographical location. FCMN [15] is another solution to handle such SSMS problems. FCMN saves energy consumption by merging multiple shortest paths for multiple sinks. It is necessary to choose the next hop nodes as the relaying nodes based on a hop counter vector (HCV). A simulation showed that FCMN outperformed SEAD in term of energy consumption. However, the next hop selection of FCMN is an ad hoc method. On the contrary, the next hop selection of the proposed method in this paper is extended from the well-known Set Cover algorithm [16]. In order to further improve the routing performance, a pruning vector (PV) is introduced in this paper. The PV removes the redundant transmission paths, and further lowers the energy consumption. As a self-contained algorithm, Hop count based routing (HCR) also provides a maintenance mechanism to handle the consequence of faulty nodes. Aid-TREE (A-TREE) is adopted to facilitate a restricted flooding to correct the inaccurate HCR due to node failures. Moreover, Lazy-Grouping is proposed to enhance the robustness of HCR under many nodes failure.

The HCR algorithm requires 1-hop neighbor information to support the routing decision. Hence, neighbor information exchange is the main overhead of HCR. Note that this overhead exists in most of the routing methods.

Since the sensor networks will be deployed on a large scale and each sensor node is powered only by the batteries, routing simplicity is our primary design requirement. Internet-like routing schemes with many states and large routine tables do not work well with sensor networks. Traditional on-demand flooding for mobile ad hoc networks scales poorly with the number of nodes and requests. On the contrary, HCR makes a minimal amount of assumptions about radio quality, presence of GPS, and other factors. HCR constructs a forwarding tree with minimum delay. HCR is simple and scalable with minimal required state, a lower control overhead, and a smaller routing table (HCV). Tree-based routing usually is very fragile in terms of node failures. Therefore, HCR has the maintenance mechanisms to handle node failure. Even though hop-count based routing protocols in mobile ad hoc networks such as Ad-hoc On-demand Distance Vector routing (AODV) [21] and Destination-Sequenced Distance-Vector routing (DSDV) [22] have already been discussed, AODV and DSDV assume that the mobile nodes will communicate with all nodes in the networks. Therefore, the internet-like routing table maintained in each node should contain hop count information for all destination nodes. If a hundred thousand nodes are deployed in a network which is a general assumption of a sensor network deployment, the size of the resulting routing table is huge. Furthermore, AODV and DSDV are designed to support the mobility. Thus, up-to-date neighbor node information is required to flood to all nodes on either periodic or on-demand basis to ensure the accuracy of routing information. Since sensor nodes have limited computing power, memory sizes, and communication bandwidth, they don't have the resources to maintain a huge routing table and constant flooding overhead. Therefore, we take the advantage of the unique characteristics of the SSMS model in sensor networks. The hop count vector in our HCR is much smaller due to a much lower number of sinks compared with the total number of sensor nodes. In this paper, we assume that the sensor nodes are stationary. The node failure is the only cause of inaccurate hop count information. Therefore, HCR has the maintenance mechanisms to handle node failure. These mechanisms are proven to be efficient and robust through simulation.

The rest of this paper is organized as follows. Section 2 presents the HCR algorithm, and describes how to determine the efficient transmission path and

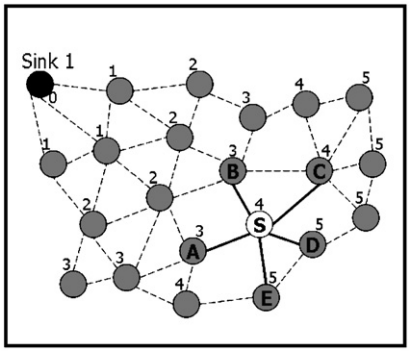

Fig. 5. Example of HCV_1.

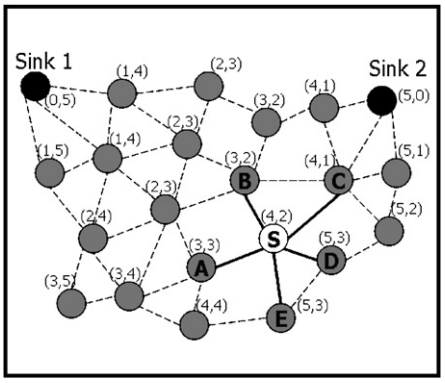

Fig. 6. Example of HCV_2.

conduct path aggregation. Section 3 presents the maintenance and error correction scheme of HCR, and describes the use of A-TREE to correct inaccurate HCV resulting from failed nodes. Section 4 discusses the robustness of the HCR algorithm. The proposed Lazy-Grouping algorithm is introduced and combined with HCR to produce a more robust HCR algorithm called LG-HCR, which is also compared with the original HCR. Section 5 presents the simulation results and compared HCR with other approaches. Finally, conclusions are presented in Section 6.

2. HCR algorithm

This section describes three main techniques of HCR, namely HCV, extended set cover (ESC), and prune vector (PV). In this paper, we assume that the sensor nodes are lack of mobility, which is very common in many sensor network applications. The proposed method assumes that each device has the same transmission range, so the connection between any two nodes is bidirectional.

2.1. Hop count vector

Each node must obtain the hop count vector only once. The following statement defines HCV of node X.

$HCV_n(X) = \{ (X_1, X_2, \ldots, X_n) |$ the hop count vector of node-X, which consists of n components, and n denotes the total number of sinks. Each component X_k, called a hop count value, indicates that the minimum hop distance from sink-k to itself is X_k hops away$\}$.

Hence, if the network topology has only one sink, then the hop count vector has only one component. Fig. 5 illustrates an example of HCV_1 for Sink-1, which demonstrates that each node holds one value, namely the hop distance from sink 1 to itself. Each node applies flooding to obtain its hop distance from each sink, and maintains a record of this value in HCV. It is similar to Beacon Vector in beacon vector routing (BVR) [6], proposed by Fonseca et al., for SSSS routing in wireless sensor networks. BVR assigns coordinates to nodes based on the vector of hop count distances to a small set of beacons, and then defines a distance metric based on these coordinates. A routing path with the topology with HCV_1 is very simple to obtain. In Fig. 5, the hop count value of Source is 4 when it transmits packets to Sink-1. Therefore, the next hop has a hop count value of 3. Nodes closer to the sink have smaller values. The rest of the hop count values can be deduced by analogy. Path aggregation will not be performed in this simple case. HCR focuses on the case of HCV_n, where $n \geqslant 2$. The HCR algorithm must be adopted when handling complex cases as in Fig. 6, which illustrates a case of HCV_2 (1, 2).

2.2. Next hop selection

The HCV of a node denotes the hop distance from itself to each sink. In a connecting wireless sensor network with n sinks, node-X is assumed to have a neighbor node-Y, and the HCVs of the two nodes are $(X_1, X_2, X_3, \ldots, X_n)$ and $(Y_1, Y_2, Y_3, \ldots, Y_n)$, respectively. The following lemmas are the properties of HCV and are proven in Appendix.

Lemma 1. *In a connected wireless sensor network with n sinks, if node-X is not a sink node, there exists at least a neighbor node-Y that* $X_k - Y_k = 1$, *for any k in the range* $[1, n]$.

HCR exploits this property to guarantee that HCR always chooses a group of nodes that are one step closer to the sinks.

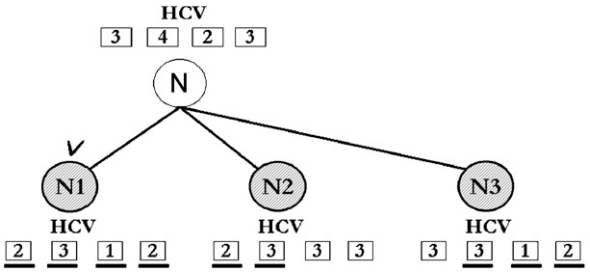

Fig. 7. Example of next hop selection.

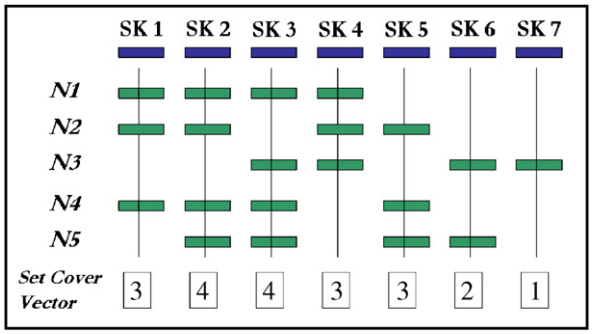

Fig. 8. Example of extended set cover.

A good-node (GN) of node X is defined as follows:

GN(X) = {node-Y|for a node-X which HCV(X) = (X_1, X_2, …, X_n), if there exists a neighborhood node-Y, which HCV(Y) = (Y_1, Y_2, …, Y_n), and $X_k - Y_k = 1$, for any k where $1 \leqslant k \leqslant n$.}

Lemma 2. *For a node-X, if there exists a neighbor node-Y that is a* GN(X)*, then node-Y, as the next hop, will be one step closer to all sinks.*

For example, in Fig. 7, node-$N1$ is a GN(N), since each component of (2,3,1,2) is less then (3,4,2,3). Hence, we choose node-$N1$ as the next hop, which is one step closer to each sink. However, in some cases no GN(N)'s exist. Hence, additional neighbor nodes may be selected as the next hop in order to guarantee minimum delay for all sinks.

Lemma 3. *For a node-X, there exists a set of neighbor nodes S, and* X_k, $-S_k = 1$ *for any k in the range* [1, *n*]*, where* $S_k \in$ HCV *of S, that are one step closer to all sinks.*

However, increasing the number of next hop nodes will increase power consumption. Therefore, the Extended Set Cover algorithm is proposed below to choose a minimum number of nodes as the next hop. This problem can be transformed perfectly into set cover problem [16].

2.3. Extended set cover

The conventional Set Cover algorithm [16] is modified and called the ESC scheme. ESC can efficiently select the nodes of next hop. A new dimension of set cover vector (SCV) is adopted to determine which neighbor node has the highest priority as the next hop. We define SCV of node X as followings:

SCV$_n$(X) = {(X_1, X_2, …, X_n)| each node holds a set cover vector which consists of n components, and n denotes the total number of sinks. Each component has a set cover value, e.g., X_k, indicating the total number of node-X's neighbor nodes which approach one more hop toward sink-k.}

The SCV is constructed by summing up the total number of neighbor nodes, which can advance one step closer to a sink for all sinks. That information can be obtained by exchanging the HCVs within neighbor nodes. If a set cover value equals 1, the corresponding neighbor node which contributes alone to this value has the highest priority. Because only one node can approach that sink, it must be chosen as the next hop in order to guarantee minimum delay. If no set cover value is 1, then the well-known Set Cover algorithm [16] is applied. An example of the operation of ESC is as follows. Fig. 8 illustrates seven sinks (SK1–SK7) and five neighbor nodes ($N1$–$N5$) for the candidates of next hop nodes. The figure shows that $N1$ approaches SK1, SK2, SK3, and SK4. N2 approaches SK1, SK2, SK4, and SK5, and so on. SK1 has a value of 3, since it is approached by three nodes, $N1$, $N2$, and $N4$. $N3$ has the highest priority as the next hop because SK7 has a set vector value of 1, meaning that only $N3$ can approach one hop closer to SK7. After choosing $N3$ as the next hop, $N3$'s corresponding elements which approach one hop closer to SK3, Sk4, SK6, and SK7 in SCV are updated to zero, indicating that those sinks have been approached by $N3$, as shown in Fig. 9. Therefore, the SCV value of each sink is checked again, and the original Set Cover algorithm is then applied to choose either $N2$ or $N4$. In Fig. 9, either $N2$ or $N4$ can cover the residual sinks, and we choose the node with more residual energy. This procedure can avoid choosing the same nodes as the next hop all the time. Thus, the power consumption can be balanced among all nodes. Let us assume that $N4$

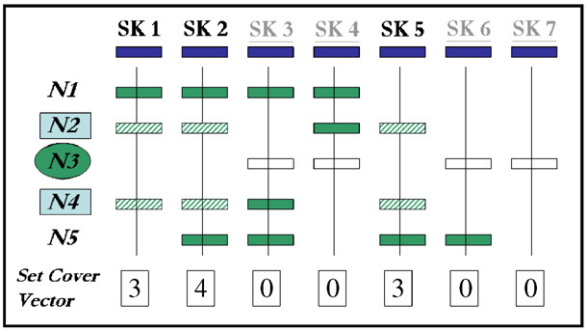

Fig. 9. Example that after N3 is selected.

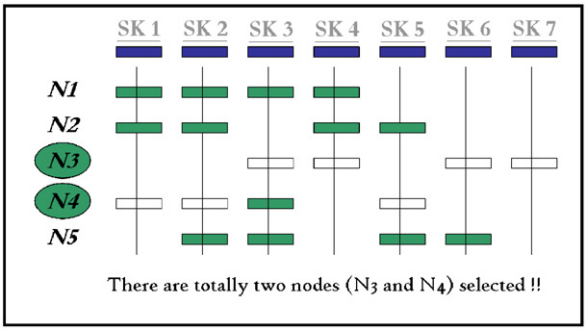

Fig. 10. Final result of the next hop selecting procedure.

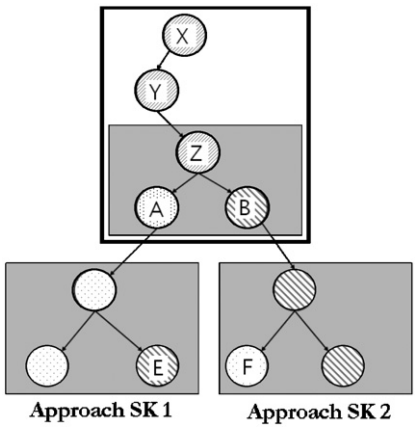

Fig. 11. The next hop selection without PV.

has more residual energy than $N2$. Finally, in Fig. 10, two nodes ($N3$ and $N4$) are selected as the next hop nodes in this selecting procedure. This technique is more efficient than the original Set Cover, which selects three nodes ($N1$, $N2$, and $N3$) in the same example. This following property guarantees HCR works.

Lemma 4. *In a connected wireless sensor network with n sinks and the source node-X, HCR algorithm guarantees that messages could reach each sinks from node-X with minimum number of hops.*

2.4. Prune vector

ESC tries to find a set of nodes, which can advance to all sinks by one hop. As shown in Fig. 11, after source node-X approaches two sinks SK1 and SK2 by two good nodes node-Y and node-Z, ESC finds a set of nodes node-A and node-B to approach two sinks SK1 and SK2 respectively. In Fig. 11, the nodes with slanted lines means that they approach SK2, and nodes with dots means that they approach SK1. However, later on as demonstrated in Fig. 11, node-E in the subtree of node-A is selected by ESC to approach the sink SK2, which has been approached by the subtree of node-B. It makes node-E redundant. The same situation occurs for node-A and node-F. This violates the objective of the study, which is to choose the minimum number of nodes as the next hop. Therefore, a pruning algorithm based on the PV is proposed.

Like HCV, the PV comprises n components, where n is the number of sinks. The component of the vector has two possible values, 1 or 0, denoting whether sinks are pruned. If the value of a component of the PV is 1, it indicates that the corresponding sink has been approached by another branch of the forwarding tree as in Fig. 11. Therefore, the algorithm does not have to choose the next hop to approach the corresponding sink. Fig. 12 shows an example of PV, in which node-Z picks two nodes using ESC as the next hop. Node-A

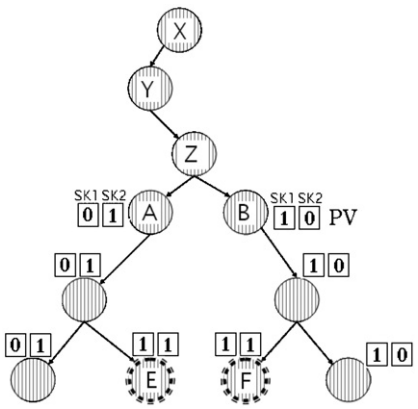

Fig. 12. Example of PV.

approaches SK1, and node-B approaches SK2. Hence, node-Z sets the pruning bit of SK2 to 1, and notifies node-A. At the same time, it sets the pruning bit of SK1 to 1 and notifies node-B. PV has the property of inheritance, which means that if one node prunes some sinks, then all its children also prunes the same sinks as illustrated in Fig. 12. Therefore, if one node has all its values of PV equal to 1, the node can be pruned such as Node-E and Node-F in Fig. 12; since whichever sinks the node approaches, they all have been covered by other forwarding branches. Thus, HCV with PV can further reduce redundant relaying nodes to save power.

3. HCR maintenance scheme

This section discusses the maintenance scheme of HCR, including the detection of a failure node, and the determination of a flooding scope in order to update inaccurate HCVs due to a node failure. When a node fails, the HCR has the additional overhead to correct the inaccurate HCVs of nodes. This paper proposes a tree-structure algorithm called Aid-TREE to achieve the great reduction of overhead to 5% of nodes comparing with traditional 100% of nodes using full-scale flooding. The following subsection expounds these techniques.

3.1. Flooding scope

A failure may leave some nodes with inaccurate HCVs, and result in the HCV correcting algorithm being applied. The HCV of each node is obtained when hop count information is exchanged by flooding after deploying the sensor network. Hence, flooding can be performed again to obtain the correct HCV. However flooding is very expensive, and only a few inaccurate HCVs need to be corrected. Therefore, an efficient correction mechanism is proposed to correct the inaccurate HCVs in a limited area, called flooding scope. Flooding scope is the maximum distance of flood from a failed node. We define the flooding scope as follow:

Flooding scope(X) = {{node-Y}|node-Y is the node with a path to node-X and with the inaccurate HCV due to node-X's failure.}

The HCVs of the nodes within the flooding scope must be corrected, while nodes outside the flooding scope have accurate HCVs. Therefore, the objective is to determine the flooding scope and perform limited flooding.

The terminal nodes (TN) are defined as follows:

TN(X) = {{node-Z}| while a node-X fails, there exists a link between node-Z and node-Y, where node-Z is outside the flooding scope and node-Y is inside the flooding scope.}

The goal can be reduced to just finding TNs, from which limited flooding can be applied and inaccurate HCVs can be updated. This is because the maximum flooding scope is bounded by TNs. Aid-TREE (A-TREE) is proposed to identify the TNs, and is explained in the next subsection.

3.2. Aid-TREE

We define the beginning node (BN) as follows:

BN(k) = {{node-B}|node-B is a node directly adjacent to a failed node and whose kth component of HCV is inaccurate.}

Since the node adjacent to the failed node could find itself as a BN, the BN is the root of Aid-TREE. The correcting mechanism will repair each inaccurate component by broadcasting the information containing the value of its inaccurate component plus one, as illustrated in Fig. 13. The nodes then receive the message, and check its relative component against the message. If they are equal, the value of the components is increased by one and is forwarded to its neighbors. Otherwise, the node sets itself as TN, as illustrated in Fig. 14. In the second phase, TN backwardly notifies the node if it has larger component value than TN, and the node sets itself as a new TN like the shaded node in Fig. 15. Otherwise, the TN remains as a TN. The procedure is repeated until all TNs are discovered. Fig. 15

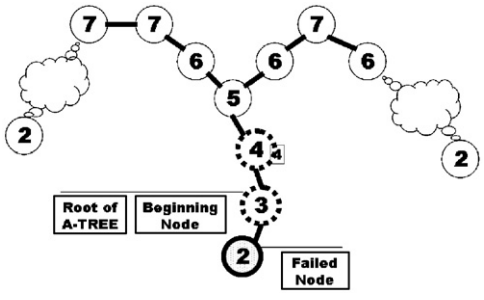

Fig. 13. Operation of finding an A-TREE.

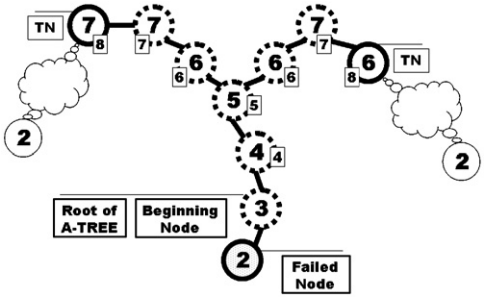

Fig. 14. The first phase of finding an A-TREE.

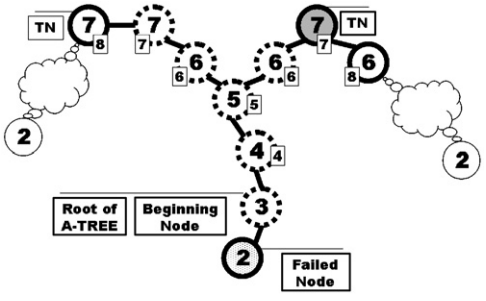

Fig. 15. The second phase of finding an A-TREE.

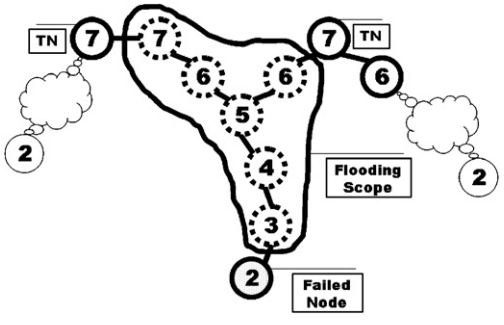

Fig. 16. A final A-TREE.

displays the final result of A-TREE, and Fig. 16 shows the flooding scope bounded by the TNs. The restricted flooding is then performed starting from all TNs, and finally the topology has the correct HCV information again.

4. Robustness improvement

This section explores the impact of a failed node. Node failures may occur at any time, causing sensor nodes to have wrong HCVs. Fortunately, it is not necessary to correct HCVs in most cases where node failures occur. In our simulation, only 15% of failure nodes cause inaccurate HCVs. Even through we have proposed efficient maintenance schemes, A-Tree and limited flooding, to correct the inaccurate HCVs. In this paper, we propose an additional improvement in robustness with the clustering technique, which we call Lazy Grouping (LG). LG is implemented to cooperate with original HCR (called LG-HCR) to further improve the robustness. Fig. 17 shows an example of the LG method. To choose appropriate cluster heads from a network to group the nodes, each node must be prioritized. Many methods are available for setting node priorities [17–19]. This paper adopts the simplest method, in which the priority of a node is determined from its remaining energy. The node with the highest priority initially acts as a group leader, and it notifies its neighborhood within a two-hop range. These neighbors receive the message, and accurately set the corresponding group leader. Repeating the process, a group leader is generated from the nodes that have not been grouped, and notifies the related group members, as shown in Fig. 17. The whole network applies the HCV assignment to each group; therefore, the nodes within the same group will hold the same hop count value. Fig. 17 shows the result of conducting the LG method. When we route data from a source to multiple sinks in the Lazy Groups, we apply the same HCR algorithm to the virtual topology derived from the original network topology as shown in Fig. 17. Instead of directly forwarding data to the next hop, a node should forward data to its group leader first. The group leader will be responsible to forward data to the next group according to the rule of HCR until the destinations have been reached. For example, in Fig. 17, node-*D* in the group with its HCV labeled as 2 (namely group-2) would like to transmit packets to Node-*C* with the HCV labeled as 3 (Group-3). Node-*D* first queries its group leader about how to reach group-3. Then node-*D* gets the routing information from its group leader and transmits the packets to group

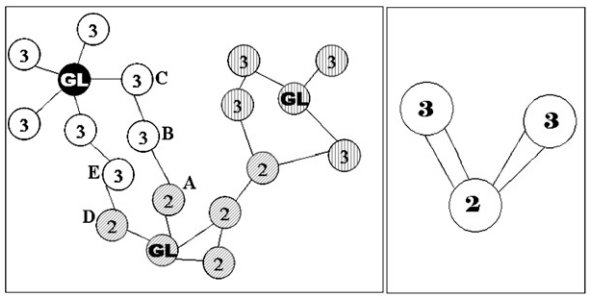

Fig. 17. Example of Lazy Grouping.

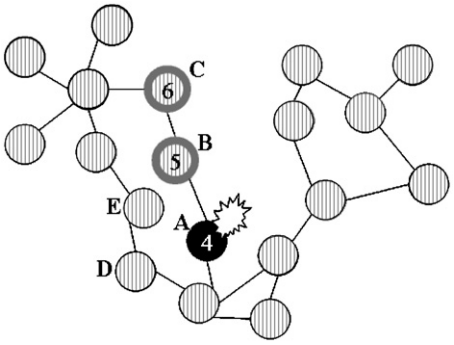

Fig. 18. Network topology without LG.

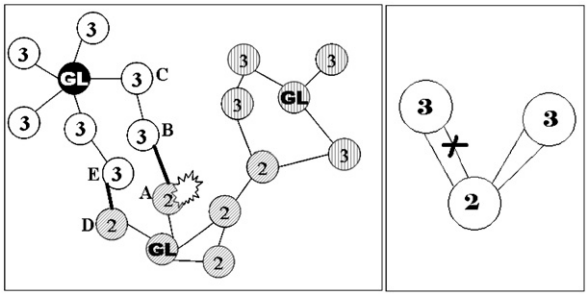

Fig. 19. Network topology with LG.

lead of Group 3, and then to node-*C*. Therefore, it will increase the transmission delay comparing with the original HCR.

The reason why LG improves robustness is explained below. Fig. 18 illustrates the same topology as in the previous example without LG and with HCV assignment. Node-*A*, node-*B*, and node-*C* have HCVs of 4, 5, and 6, respectively. If node-*A* has failed, then node-*B* and node-*C* both have inaccurate HCVs. Therefore, these two nodes should run correcting algorithms and update the inaccurate HCVs. On the contrary, in Fig. 19, the original node-*A*, node-*B*, and node-*C* have HCV values of 2, 3, and 3, respectively. The left-hand side of Fig. 19 represents the LG topology, which can be regarded as the virtual topology as in the right-hand side. The LG topology has high connectivity among neighbor groups, because each group has lots of links to its neighbor groups. Therefore, the HCV has less effect due to the failed nodes, as shown in Fig. 19. Nodes in the group with a HCV labeled as 2 may reach node-*E*, which the HCV labeled as 3, through node-*D*, and irrespective of the failed node-*A*. Hence, LG can reduced the frequency of HCV update and improve the robustness of HCR routing in sensor networks.

5. Simulation results and performance analysis

The performance of a routing algorithm is usually evaluated according to connectivity, energy consumption, throughput, and robustness [20]. To evaluate the HCR protocol, three main categories, energy consumption, robustness, and delay were adopted in this study to compare HCR with BF and FCMN [15] methods. These routing algorithms were simulated using C++ programs. The task of a source sensor node is to gather data and deliver it to multiple clients who are interested in the data. In our simulation, we assume that traffic is not heavy, which is a reasonable assumption for the transmission properties of sensor nodes. Hence, the influence of contention and collision is negligible in our simulation.

5.1. Performance metrics

We define the performance metrics as follows:

Energy consumption: The total number of transmissions was adopted to evaluate the energy cost. The numbers of transmissions in HCR, BF and FCMN were compared by varying the network topology. The number of relay nodes equals the total number of transmission. Many transmissions imply a high-energy cost.

Robustness: The network topology had 400 nodes with IDs assigned from 1 to 400 and the topology size varying from 3400×3400 to 2200×2200. The original HCR was compared with LG-HCR in terms of robustness. The robustness of algorithm was evaluated using the following parameters.

Total number of NTR: Each node k where k was looped from 1 to 400 was made to fail in turn, and the number of times each failed node resulted in inaccurate HCVs among its neighbors was recorded. If the failed node affected its neighbors' HCVs and the topology became 'need to repair' (NTR), then the total number of NTR was increased by one. A small NTR value indicates a high robustness.

Average number of CN (flooding scope): Each node k where k was looped from 1 to 400 was made to fail in turn, and the number of 'corrected nodes' (CN) was the summation of number of nodes with inaccurate HCVs that resulted from a failed node for 400 nodes. The average number of CN is the number of CN over the number of NTR. A smaller average number of CN indicates a less maintenance cost.

The improvement of CN: We define the improvement of maintenance cost using the saving of CN in LG method over the original HCR as the improvement of CN. The equation is as follows:

Delay: Delay is defined as the number of hops the packet transmits from the source to the destinations. Note that the effect of contention and collision is not considered here according to our previous light traffic assumption. Since HCR guarantees minimum delay, we compare only the delays of the original-HCR and LG-HCR. This performance evaluation shows the extra delay cost using the LG method.

5.2. Source and destination scenarios

The performance results were obtained using different source and destination (S/D) scenarios, namely Corner, Full-Corner, Line, Block, and Cycle, as illustrated in Fig. 20. We use these different S/D types to replicate the real network scenarios including two extreme cases. One is that the sinks are uniformly distributed around the circle which we call cycle; the other is that the sinks are distributed in a confined area which we call block. The description of the S/D types is as follows:

Corner: the source is in area A, and the sinks are in areas D, E and F.

Full-Corner: the source is in area A, and the sinks are in areas C, D, E, F and G.

Line: the source is in areas A, B and C, and the sinks are in areas E, F and G.

Block: the source is in area B, and the sinks are in area F.

Cycle: the source is in area I, and the sinks are in areas A, B, C, D, E, F and G.

The simulation of path aggregation shows that sinks should not be uniformly distributed around the circle, and should be distributed in a confined area. In the worst case, aggregating the path produces no benefit, and the HCR aggregation method deteriorates to BF method.

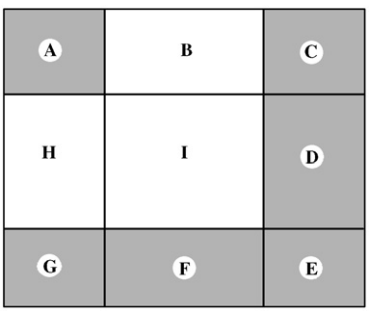

Fig. 20. Full-Corner 4/8/16 topology.

The performance metrics under the simulation environment of different S/D types is described below.

Energy consumption: The following scenarios were adopted to evaluate the energy consumption: Block-4; Corner-4; Line-4, 8, and 16; Full-Corner-4, 8, and 16; Cycle-4, 8, 16, and 32, where 4, 8, 16, and 32 indicate the number of sinks. The average node density was varied by depositing different numbers of nodes in the topology with a size of 3000×3000 units. Note that the size of network was fixed in this simulation.

Robustness: The robustness was simulated using Full-Corner-4, and the average node density was varied by changing the topology size from 3400×3400 to 2200×2200. The number of nodes was fixed at 400 in this simulation.

Delay: The delay was evaluated using Full-Corner-4, and the average node density was varied by depositing different number of nodes in the topology with a size of 3000×3000 units. The topology size was fixed in this simulation. Because of the Full-Corner type, the difference of delay between HCR and LG-HCR could easily be observed.

5.3. Performance result and analysis

The performance of HCR was evaluated with different network parameters, including S/D types and node densities. We plot the average value of one hundred simulations for each set of network parameters. Simulation results indicate that HCR is better than other methods in all respects.

Figs. 21–23 present the variation in the number of transmissions with node density when HCR, BF and FCMN are adopted, respectively. These results show that the energy cost of FCMN grows as the node density increases, since the choices of possible paths also increases, and the redundant paths are not adequately pruned. The energy cost of BF grows as the number of sinks increases, because BF transmits the packet to each sink along many paths, and does not perform the path aggregation. However, the energy cost of BF decreases as the node density increases, since the choice of possible paths also increases. The method therefore always has a good chance of choosing a better next hop. Figs. 24–27 indicate the same argument with

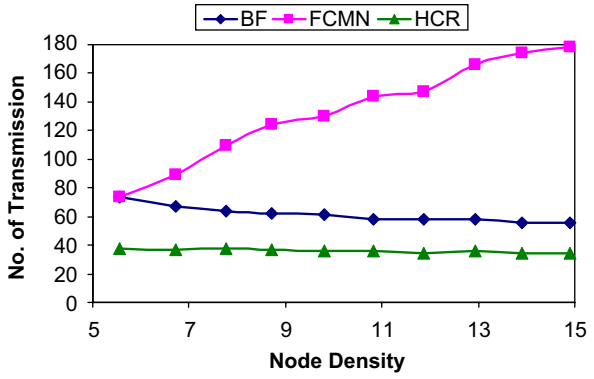

Fig. 21. Full-Corner 4: Node density vs. number of transmission.

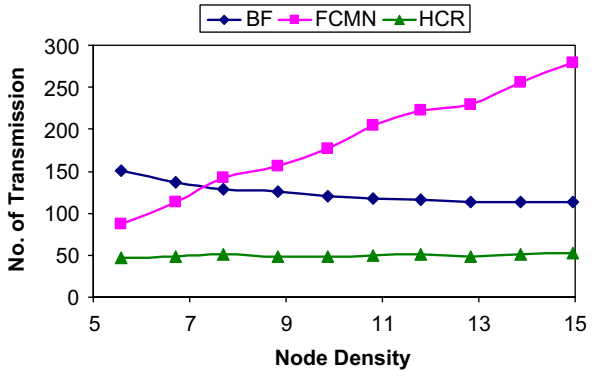

Fig. 22. Full-Corner 8: node density vs. number of transmission.

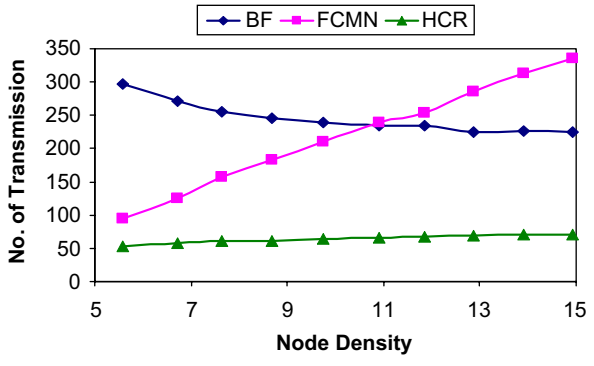

Fig. 23. Full-Corner 16: Node density vs. number of transmission.

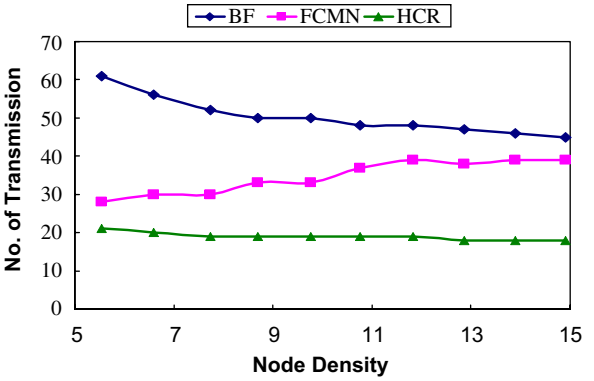

Fig. 24. Block 4: node density vs. number of transmission.

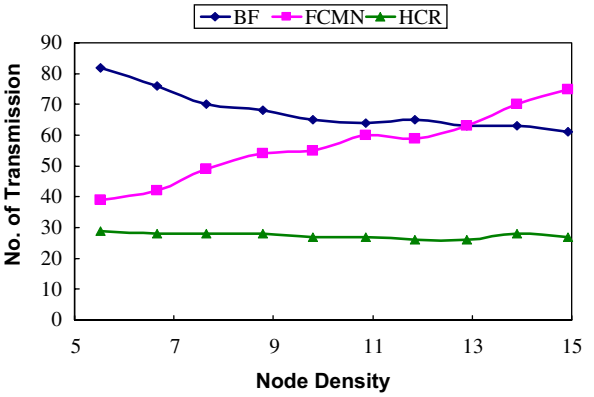

Fig. 25. Corner 4: node density vs. number of transmission.

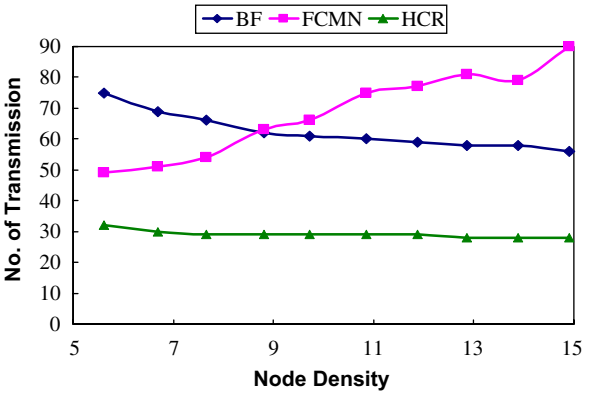

Fig. 26. Line 4: node density vs. number of transmission.

varying network topologies. Furthermore, Fig. 24 shows that the benefit of path aggregation is more apparent in the topology with sinks clustered together than the topology with circumfluent sinks. In the worst case, Fig. 27 shows that the HCR method has similar number of transmission with the BF method, and therefore aggregating the path almost produces no benefit. Overall, the energy cost of HCR is more stable and less costly than other methods.

Fig. 28 shows that the energy consumption using LG-HCR is slightly higher than using the original

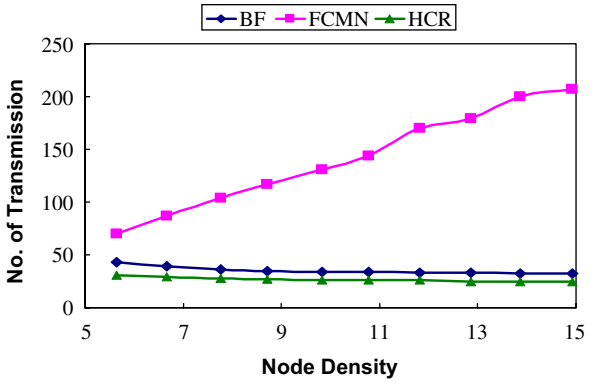

Fig. 27. Cycle 4: node density vs. number of transmission.

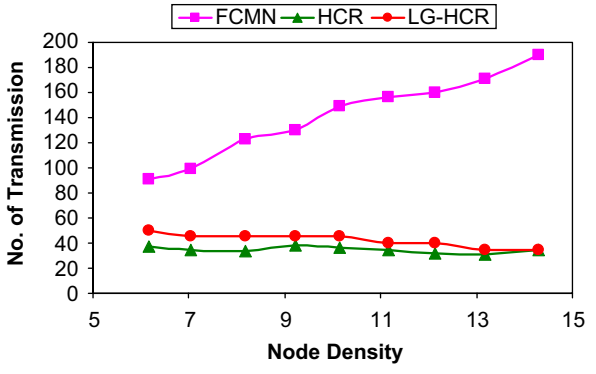

Fig. 28. Full-Corner-4: energy cost using LG-HCR.

HCR due to more relay nodes to transfer data within groups.

Fig. 29 shows that LG-HCR has better robustness in terms of the total number of NTR. It is also observed that the higher node density has fewer number of NTR. In other words, topology with higher node density is more robust. In this simulation, the 400 nodes were evenly distributed within the area with sizes from 3400×3400 to 2200×2200. There are 400 simulation runs. In the kth run, we let node-k fail and checked whether it caused the remaining nodes to have inaccurate HCVs. If the failed node affected its neighbors' HCVs and the topology became NTR, then we increased the number of NTR. In HCR algorithm, in average only 15% of nodes out of 400, (i.e. 60 nodes) when they fail, will activate the HCV correction procedure. However, in LG-HCR algorithm, only about 7% of nodes out of 400 (i.e. 28 nodes) are NTR in average. Fig. 30 displays the average flooding scope of the two methods under different node density. Both methods have a similar number of nodes' HCVs that need to be corrected. In average only about 5% of 400 nodes (i.e. 20 nodes) need to be updated with the correct HCVs for each NTR case. Restricted flooding can save more energy compared with full-scale flooding because the flooding scope is relative small. These results can be easily expounded. For a Node-X, and its neighbor Node-Y, their HCVs of the two nodes are $(X_1, X_2, X_3, \ldots, X_n)$ and $(Y_1, Y_2, Y_3, \ldots, Y_n)$, respectively. It has a property as $|X_k - Y_k| \leqslant 1$, for any k in the range $[1, n]$. It implies that there are only three kinds of node-X's neighbors where the difference of hop count value between node-X and its neighbor are 1, 0, and -1. If node-Y fails, it makes the kth component of HCV of node-X inaccurate only when $Y_k < X_k$ and node-Y is the only neighbor of node-X that the kth component is smaller than node-X. Since nodes in sensor networks usually are deployed with high density, their average number of neighbors is much larger. They usually have few chances to conform to the above two conditions,

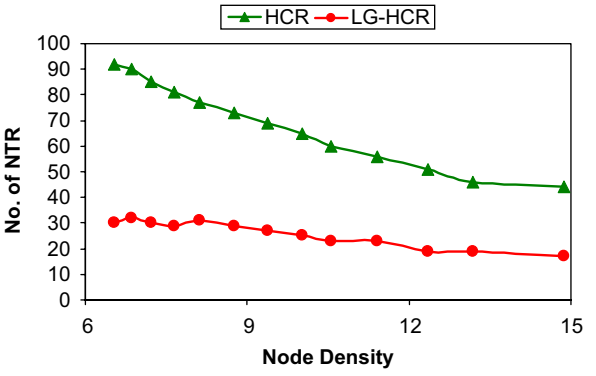

Fig. 29. Robustness of HCR and LG-HCR.

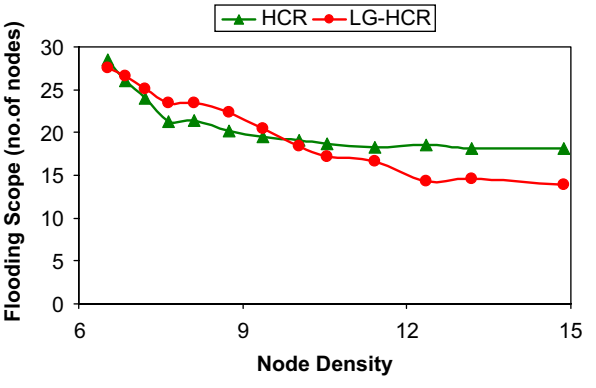

Fig. 30. Flooding scope of HCR and LG-HCR.

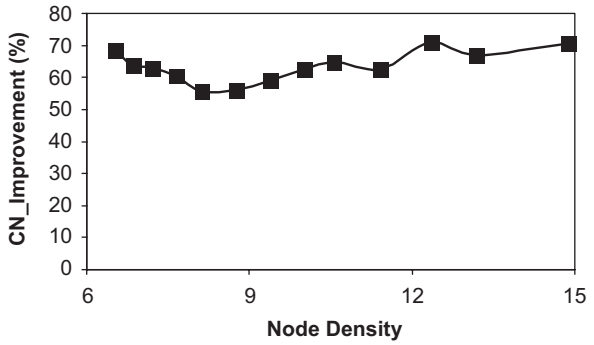

Fig. 31. Improvement in maintenance cost using LG-HCR over HCR.

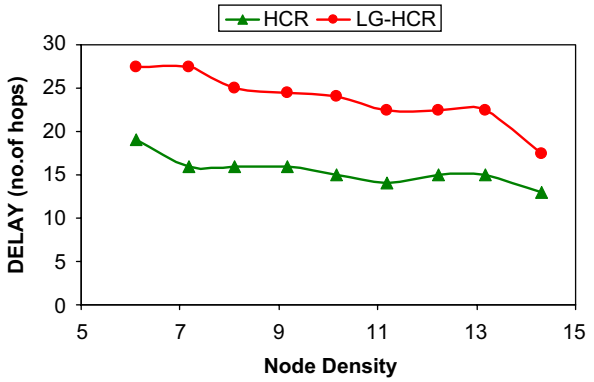

Fig. 32. Node density vs. transmission delay.

especially under higher node density. Therefore, the percentage of NTR is small. Even through NTR does happen sometimes; the flooding scope is very small due to analogous reasons described above. Fig. 31 shows that LG-HCR produces a 60% improvement over HCR in the total number of CNs, indicating that LG-HCR is much more robust than HCR.

Fig. 32 shows the extra delay cost by LG. This is because LG has to relay packets through group leaders. However, the original HCR has the same delay with BF, because HCR combines the shortest paths from one source to each sink. Finally, Fig. 33 comprehensively compares HCR and LG-HCR in terms of delay, robustness and energy consumption, and shows that LG-HCR is significantly more robust than HCR, but also has a longer delay and slightly higher energy cost.

6. Conclusion

This paper has modeled and analyzed the performance of HCR in wireless sensor networks.

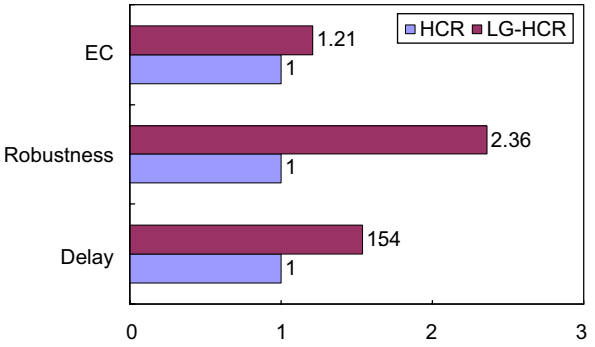

Fig. 33. Comprehensive comparison between HCR and LG-HCR.

HCR is simple, scalable, and robust against node failure. It provides the minimum delay and energy efficient SSMS routes. The objective of the proposed algorithm is to choose the appropriate nodes for the next hop and to perform path aggregation. The end-to-end transmission delay of our method is as short as the brute-force method theoretically. Through simulation, we find that the power consumption of our method is more efficient comparing with other methods. The impact of failed nodes was studied, and LG was proposed to improve the robustness of HCR. In addition to providing the maintenance algorithm, the proposed algorithm performs restricted flooding to handle the effects caused by the faulty nodes. A major result of HCR is that only about 15% of nodes out of 400 are NTR. Furthermore, the LG-HCR algorithm improves NTR to about 7% of nodes out of 400. The flooding scopes of the two methods are similar. About 5% of 400 nodes are needed to be updated with correct HCVs in each NTR case. The performance of restricted flooding is compared with using full-scale flooding. HCR is a complete solution comprising routing, grouping and maintenance. HCR performs significantly better than other algorithms, since it simultaneously considers energy cost and transmission delay.

Acknowledgement

Authors like to thank Dr. Chao-Ming Liu for his valuable comments.

Appendix

Lemma 1. *In a connected wireless sensor network with n sinks, if node-X is not a sink nods, there exists*

at least a neighbor node-Y that $X_k - Y_k = 1$*, for any k in the range* [1, *n*].

Proof. Since it is a connecting network, for each sink-*k*, we can find a shortest path from the sink-*i* to node-*X*. Since the shortest path must go through one of node-*X*'s neighbor nodes (node-*Y*), to reach node-*i*. If the *k*th element of HCV of node-*Y* is equal to n, then the *k*th element of HCV of node-*X* is equal to $n+1$ based on the definition of HCV. Hence, $X_k - Y_k = 1$ for any k in the range $[1, n]$. □

Lemma 2. *For a node-X, if there exists a neighbor node-Y that is a* GN(*X*)*, then node-Y, as the next hop, will be one step closer to all sinks.*

Proof. It can be proved by the definition of good-node and HCV. □

Lemma 3. *For a node-X, there exists a set of neighbor nodes S, and* X_k, $-S_k = 1$ *for any k in the range* [1, *n*]*, where* $S_k \in$ HCVs *of S, that are one step closer to all sinks.*

Proof. For a node-*X*, the HCV of node-*X* is (X_1, X_2, X_3, …, X_n). According to Lemma 1, we could find a set of neighbor nodes *S*, namely node-S_1, node-S_2, …, and node-S_m. The HCV of S_i is ($S_{i,1}$, $S_{i,2}$, $S_{i,3}$, …, $S_{i,k}$, …, $S_{i,n}$) and $X_k - S_{i,k} = 1$, for each k, where $k \in [1, n]$. According to Lemma 2, these nodes are one step closer to the sinks than node-*X*. □

Lemma 4. *In a connected wireless sensor network with n sinks and a source node-X, HCR algorithm guarantees that messages could reach each sinks from node-X with minimum number of hops.*

Proof. According to Lemma 3, for a source node-*X* the HCR choose a set of nodes using set cover as the next hop, it guarantees that these intermediate nodes are one step closer to all sinks. Finally, we could find routing paths to reach each sinks from node-*X* with minimum number of hop. □

References

[1] G.J. Pottie, W.J. Kaiser, Wireless integrated network sensors, Commun. ACM 43 (5) (May 2000) 51–58.

[2] D. Estrin, L. Girod, G. Pottie, M. Srivastava, Instrumenting the world with wireless sensor networks, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001), Salt Lake City, Utah, May 2001.

[3] R. Musaloiu, A. Terzis, K. Szlavecz, A. Szalay, J. Cogan, J. Gray, Life under your feet: A wireless soil ecology sensor network, in: Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets 2006), Cambridge, MA, May 2006.

[4] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, in: ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom '99), Seattle, Washington, August 1999.

[5] W.R. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive Protocols for information dissemination in wireless sensor networks, in: Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99), Seattle, Washington, August 15–20, 1999, pp. 174–185.

[6] R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, I. Stoica, Beacon vector routing: scalable point-to-point in wireless sensornets, Technical Report IRBTR-04-12, Intel Research Berkeley, May 2004.

[7] B.J. Bonfils, P. Bonnet, Adaptive and decentralized operator placement for in-network query processing, in: Proceedings of the Second International Workshop on Information Processing in Sensor Networks (IPSN '03), Palo Alto, CA, April 2003, pp. 47–62.

[8] V. Chowdhary, H. Gupta, Communication-efficient implement-action of join in sensor networks, in: Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA '05), Beijing, China, April 2005, pp. 447–460.

[9] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, TAG: a Tiny AGgregation service for ad-hoc sensor networks, in: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, MA, December 2002, pp. 131–146.

[10] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, The Design of an Acquisitional Query Processor For Sensor Networks, ACM SIGNOD, San Diego, CA, June 2003, pp. 491–502.

[11] S. Madden, R. Szewczyk, M.J. Franklin, D. Culler, Supporting aggregate queries over ad-hoc wireless sensor network, in: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, Callicoon, NY, June 2002, pp. 49–58.

[12] H. Kim, T. Abdelzaher, W. Kwon, Minimum Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks, SenSys'03, LA, CA, November 2003, pp. 193–204.

[13] G. Jiang, G. Cybenko, Query routing optimization in sensor communication networks, in: Proceedings of the 4lst IEEE Conference on Decision and Control, Las Vegas, December 2002, pp. 1999–2005.

[14] F. Heide, C. Schindelhauer, K. Volbert, M. Grünewald, Energy, congestion and dilation in radio networks, in: Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures, 2002.

[15] C.H. Ke, et al., Merged multi-path routing: two energy-efficient strategies in wireless sensor networks, in Chinese, in: Proceedings of the National Computer Symposium (NCS), Taiwan, December 2005.

[16] D.S. Johnson, Approximation algorithms for combinatorial problems, J. Comput. Syst. Sci. 9 (1974) 256–278.

[17] L. Bao, J.J. Garcia-Luna-Aceves, Topology management in ad hoc networks, in: Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking & Computing, 2003.
[18] Y. Xu, W.-C. Lee, J. Xu, G. Mitchell, PSGR: Priority-based Stateless Geo-Routing in Wireless Sensor Networks, in: The Second IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS '05), Washington, DC, November 2005, pp. 673–680.
[19] J. Carle, D.S. Ryl, Energy efficient area monitoring for sensor networks, IEEE Comput. 37 (2) (February 2004) 40–46.
[20] R. Rajaraman, Topology control and routing in ad hoc networks: a survey, ACM SIGACT News, June 2002.
[21] C.E. Perkins, E.M. Royer. Ad-hoc On-Demand Distance Vector Routing, in: Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90–100.
[22] C.E. Perkins, P. Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, in: Proceedings of ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM '94), London, UK, August 1994, pp. 234–244.