Leveraging 1-hop Neighborhood Knowledge for Connected Dominating Set in Wireless Sensor Networks

Wenyong Wang
School of Computer Science and Engineering in University of Electronic Science and Technology of China, Chengdu, China
Email: wangwy@uestc.edu.cn

Jun Zhang, Yong Tang, Yu Xiang and Ting Yang
School of Computer Science and Engineering in University of Electronic Science and Technology of China, Chengdu, China
Email: jun.wudu@gmail.com, {worldgulit, jcxiang, yting}@uestc.edu.cn

***Abstract*—To improve the efficiency of routing and broadcast and reducing energy consumption in the process of data transmission, calculating minimum connected dominating set is always used to construct virtual backbone network in wireless sensor networks. Calculating the minimum connected dominating set (MCDS) of plane graphs is a NP-complete problem. In this paper, an algorithm leveraging 1-hop neighborhood knowledge for connected dominating set is proposed. First, the minimum forwarding set is calculated severally by each node in the entire network. Then any one node can start the process of broadcasting messages including the information of minimum forwarding set in the network. Finally, the connected dominating set of the entire network is achieved by exchanging information. The proposed algorithm aims to get a small connected dominating set, meanwhile, to minimize the consumption of energy and time. The simulation results show that the algorithm has achieved its purpose with fast convergence, low transmission traffic and reasonable size of connected dominating set.**

***Index Terms*—Wireless Sensor Networks, 1-hop, Communication Coverage, Connected Dominating Set**

I. INTRODUCTION

Constructing virtual backbone network in wireless sensor networks has a wonderful performance in improving the performance of broadcast [1], reducing reduplicate data transmission, energy saving and bandwidth saving. Constructing a virtual backbone is identical with calculating connected dominating set in graph theory [2]. The virtual backbone of the network derived from a smaller connected dominating set [3]will not only benefit the design of energy-efficient routing, but also save energy for the reason that non-dominating nodes without monitoring task could enter sleep (energy-saving) mode.

Manuscript received December 12, 2010; revised March 22, 2011; accepted April 3, 2011.

This paper is supported by Research Fund for the Doctoral Program of Higher Education of China for New Teachers No.200806141110 and project CNGI of NDR No.CNGI-09-01-07.

As calculating the minimum connected dominating set (MCDS) of arbitrary graphs is a NP-complete problem, heuristic algorithms are usually used to calculate it approximately. There are two main heuristic algorithms: centralized algorithm [4-6] and distributed algorithm [7-15]. Although the minimum connected dominating set can be calculated by centralized algorithms, it is difficult to get the entire topology structure of dynamic wireless sensor networks which is necessary for centralized algorithms. Hence distributed algorithm is adopted in this paper. Typical distributed algorithms, such as Dai's algorithm [13] and MISB algorithm [14] need two or more hops adjacent node information to calculate the minimum connected dominating set. As a result, more messages are exchanged for calculating the minimum connected dominating set, leading to more energy consumption and time delay.

To reduce energy consumption and message exchange, improve the efficiency of algorithm and get a smaller connected dominating set, a new algorithm, Leveraging 1-hop Neighborhood Knowledge for Connected Dominating Set in Wireless Sensor Networks (OHCDS), which is based on our earlier achievement OHDC algorithm [15], is proposed in this paper. There are two stages in the calculating process of our algorithm OHCDS. In the first stage each node calculates the minimum forwarding set severally. In the second stage, the node checks out if it is a dominating node by broadcasting messages including the information of minimum forwarding set, the process of which is similar to breadth-first search of graph.

The rest of this paper is organized as follows. Section 2 describes symbols used in this paper. Section 3 analyses the CDS problem from another view. Section 4 presents details of our algorithm. Section 5 presents performance evaluation. Section 6 concludes the paper.

II. SYMBOLS DESCRIPTION

Symbols used in this paper are shown in Tab.1. We assume that transmission radius of all nodes is *R*, and all

doi:10.4304/jcp.7.1.11-18

nodes are randomly dispersed over a 2-dimensional geographical region. The communication region of the node v is $A(v)$. The whole wireless sensor network region is regarded as a connected graph G with V representing set of vertices and E representing set of edges. We get $uv \in E$ if and only if vertices u, v are adjacent and $u, v \in V$.

TABLE 1. SYMBOL TABLE

Symbol	definition
G	Equation $G=(V,E)$ denotes a graph G, V is the vertices set, and E is the edges set
R	The transmission radius of node
$N(v)$	The adjacent node set of vertex v
$N[v]$	$N[v]=N(v)\cup\{v\}$
$C(v)$	The circle with v as the center and a radius of R
$A(v)$	The region of circle $C(v)$
$A(D)$	$A(D)=\bigcup_{v\in D}A(v)$, where D is a set of vertices
$Q(v)$	The set of $A(N[v])$'s boundary intersection points
$B(v)$	If $v'\in B(v)$, then $C(v')$ is a circle at $A(N[v])$'s boundary
$\overset{\frown}{ab}$	Arc $\overset{\frown}{ab}$, whose direction is clockwise from a to b
uv	The distance between node u and node v, $u,v\in V$
CDS	The connected dominating set of graph $G=(V,E)$

III. PROBLEM STATEMENT

If source node s broadcasts packet P, and all nodes in $N(s)$, which is the adjacent node of s, forward this packet after they receive it, then all 2-hop adjacent nodes of s can receive the packet P. If we can obtain a forwarding set $F(.)$ which satisfies $A((F(s)\cup\{s\}))=A(N[s])$ $(F(s)\subseteq N(s))$, then all 2-hop adjacent nodes of s can receive the packet P as long as all the nodes in $F(s)$ participate in forwarding. If all the nodes in the network calculate their own forwarding set $F(.)$, and only nodes in $F(.)$ are designated to forward this packet P, then all the nodes can receive the packet P. These nodes which forward the packet form the connected dominating set CDS.

In order to obtain a smaller CDS, first, this algorithm makes use of local communication coverage information to calculate the minimum forwarding set $F_{\min}(.)$ of every vertex (*algorithm* 1 in section IV). Then, choose a source vertex s from V, and finally we can obtain an ideal CDS by broadcasting (*algorithm* 2 in section IV).

***Lemma* 1.**

①. $A(B(v)\cup\{v\})=A(N[v])$, ②. $F_{\min}(v)=B(v)$.

***Proof*:**

①. The boundary of region $A(N[v])$ is made up of the arcs whose center forms the set $B(v)$. Assume that $A(N[v])-A(B(v)\cup\{v\})\neq\varnothing$, then, the hole formed is inside region $A(N[v])$.

Assume that vertex c is a point inside the hole. If connect vertex c and vertex v, ray vc intersects circle $C(v')$ which is at the boundary of the hole at point a (a is out of line segment $\overline{vc}$). Since line segment $\overline{vv'}\le R$, vertex v is inside or on the circle $C(v')$. Moreover, point a is on the circle $C(v')$. Then line segment $\overline{va}$ is inside $C(v')$. So each point of $\overline{va}$ is inside or on circle $C(v')$. c is a point of $\overline{va}$ means that c is inside circle $C(v')$ as well, hence point c is not inside that hole which contradicts the assumption. So the assumption is not true. So there is no hole in $A(N[v])$, and $A(B(v)\cup\{v\})=A(N[v])$ is gotten.

②. We prove by the method of reduction to absurdity. From ① of *lemma* 1, we know that $F_{\min}(v)$ must satisfy $|F_{\min}(v)|\le|B(v)|$. Assume that $F_{\min}(v)\neq B(v)$. Since $A(B(v)\cup\{v\})=A(N[v])$, we get $A(F_{\min}(v)\cup\{v\})\neq A(N[v])$ or $A(F_{\min}(v)\cup\{v\})=A(N[v])$ and $F_{\min}(v)\supset B(v)$ $(|F_{\min}(v)|>|B(v)|)$. Those two results contradict the preconditions, so $F_{\min}(v)=B(v)$.

■

From ② of *lemma* 1, we know the required $F_{\min}(v)$ is identical with $B(v)$. For convenience, we divide the CDS problem of graph G into two problems:

Problem A is how to calculate set $B(v_x)$ for each vertex $v_x\in V$ of graph G.

Problem B is how to calculate CDS of graph G start from source s.

IV. ALGORITHM IMPLEMENTATION

A Solving Problem A

Assume $d_k=\{(x_k,y_k);(u_k,v_k)\}\in Q(v_x)$ with coordinate (x_k,y_k), and it is the intersection point of circle $C(u_k)$ and circle $C(v_k)$, which are both at the boundary of region $A(N[v_x])$ (That is, $u_k,v_k\in B(v_x)$). $B(v_x)$ can be obtained by calculating $Q(v_x)$.

Here we introduce a method to calculate $Q(v_x)$. If $u_1,u_2,...,u_i\in N(v_x)$ and $u_1v_x<u_2v_x<...<u_iv_x$, then $C(u_1)$, $C(u_2)$, … , $C(u_i)$ will be added to the obtained region in turn.

In this algorithm, let v_x to be a reference vertex. The gray region in Fig.1 represents current region $A(P(u_i))$. We use u_0 to replace v_x, let the algorithm start from u_0. Assume $u_1,u_2,...,u_i\in N(v_x)$ and $u_1v_x<u_2v_x<...<u_iv_x$, thus we get current region $A(P(u_i))$ satisfies $A(P(u_i))=\bigcup_{j=0}^{i}A(u_j)$, in which $P(u_i)=\bigcup_{j=0}^{i}u_j$.

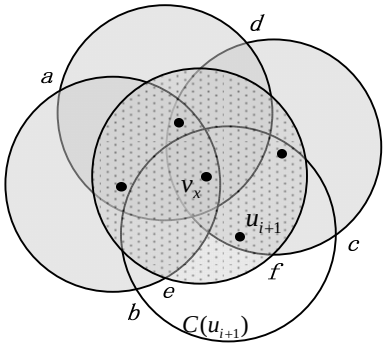

Figure 1. The process of adding circles

In Fig.1, $Q(v_x)=\{a,e,f,d\}$. $Q(v_x)$ will be updated and current region $A(P(u_i))$ will be expanded to $A(P(u_{i+1}))=A(P(u_i))\bigcup A(u_{i+1})$, if $u_{i+1}\in N(v_x)$ and circle $C(u_{i+1})$ is added to $A(P(u_i))$, then region $A(u_{i+1})$ covers vertices *e* and *f* in $Q(v_x)$, so vertices *e* and *f* are no longer at the boundary of $A(P(u_{i+1}))$. Thus *e* and *f* can be deleted from $Q(v_x)$. Simultaneously, new points *b* and *c*, the intersection points of $C(u_{i+1})$ and original region, are also at the boundary of new region. Then *b* and *c* are added into set $Q(v_x)$, and $Q(v_x)=\{a,b,c,d\}$.

According to the thought above, each time a vertex u is picked from $N(v_x)$, and $C(u)$ is added to existing region $A(P(u_i))$. If $A(u)$ covers existing vertices in $Q(v_x)$, those vertices should be deleted from $Q(v_x)$. If added $C(u)$ brings new intersection points with boundary, those new points are added into set $Q(v_x)$. This process continues until all vertices in $N(v_x)$ are added. Thus region $A(N[v_x])$ is formed, and the set of boundary intersection points $Q(v_x)$ is gotten. The set $Q(v_x)$ is just we wanted.

The correctness of this algorithm is ensured by *lemma* 2.

***Lemma* 2.**

Before adding $C(u_{i+1})$, there is no hole in the already formed region $A(P(u_i))$ which contents i added circles and $C(v_x)$.

***Proof* :**

lemma 2 can be proved by *lemma* 1 easily.

■

For convenience, the whole algorithm is divided into main algorithm and sub-algorithm.

In this algorithm, $P(u_i)$ is the set to store vertices that already added and $Q(u_i)$ is the set to store boundary intersection points of already-existed region.

***Algorithm* 1:** Calculating $B(v_x)$ of v_x

Input: set $N[v_x]$.

Output: set $B(v_x)$.

1) Initialize $A(P(u_0))=\varnothing$, $P(u_0)=\varnothing$ and $Q(u_0)=\varnothing$.

2) Put v_x into set $P(u_0)$. Add $C(v_x)$ to $A(P(u_0))$, and let $A(P(u_0))=A(v_x)$.

3) If v_x, u_1, u_2, … , u_i have been put into set P, then let $A(P(u_i))=A(v_x)\cup A(u_1)\cup A(u_2)\cup ...\cup A(u_i)$. Choose vertex u_{i+1} which is the nearest point to v_x in $N(v_x)-P(u_i)$.

4) Add $C(u_{i+1})$ to region $A(P(u_i))$, and $A(P(u_{i+1}))=A(P(u_i))\cup A(u_{i+1})$.

5) If $A(u_{i+1})$ covers some vertices in $Q(u_i)$, calculate set $Q(u_{i+1})$ by *sub-algorithm*, and then go to step 8).

6) If there is no vertex in $Q(u_i)$ covered by set $A(u_{i+1})$, then choose the nearest vertex u_j to u_{i+1} from $P(u_i)$.

7) Calculate the intersection points of circle $C(u_j)$ and circle $C(u_{i+1})$, and then put them into set $Q(u_i)$.

8) Let $i=i+1$, return step 3) until $N(v_x)-P(u_i)=\varnothing$.

9) For each $d_k\in Q$, put u_k, v_k associated with d_k into set $B(v_x)$.

10) End.

***Theorem* 1.** According to the step 3) of the *algorithm* 1,

a. adding a circle $C(u_{i+1})$ to the current region $A(P(u_i))$ will always update set $Q(v_x)$

b. added circle $C(u_{i+1})$ will intersect the boundary of region $A(P(u_i))$ at two and only two points .

***Proof* :**

(a) From step 3), it is known that, for $u_{i+1}\in N(v_x)-P(u_i)$, to $\forall u_k\in P(u_i)$, satisfies $u_{i+1}v_x>u_kv_x$, k=1,2,…,i. Thus by adding circle $C(u_{i+1})$ to $A(P(u_i))$, the circle $C(u_{i+1})$ can always intersects the boundary of $A(P(u_i))$. So set $Q(v_x)$ can always be updated.

(b) We use broken circle S' to denote the maximum scope $A(P(u_i))$ can reach. There are only two situations when adding circle $C(u_{i+1})$: ①circle $C(u_{i+1})$ intersects with circle S'; ②circle $C(u_{i+1})$ is tangent with circle S'. Fig.2 shows situation①. The gray part represents communication region of reference vertex v_x, and circle $C(u_{i+1})$ is the circle to be added. Circle *B* is in the current region. Circle $C(u_{i+1})$ intersects circle $C(v_x)$ at points *a* and *b*, and intersects circle S' at points *c* and *d*.

Assume that arc $\overset{\frown}{de}$ is the part of circle $C(u_{i+1})$ cut by circle *B* and circle $C(v_x)$, point *a* is the intersection point of circle $C(u_{i+1})$ and circle $C(v_x)$, point *e* is the intersection point of circle $C(u_{i+1})$ and circle *B*, arc $\overset{\frown}{de}$ is arc of circle *B*. If point *e* at the boundary of the current region $A(P(u_i))$, then any circle in current region $A(P(u_i))$

intersects arc $\overset{\frown}{ac}$ at a point on arc $\overset{\frown}{ae}$, thus point e is the only point on arc $\overset{\frown}{ac}$ at the boundary of region $A(P(u_i))$. Similarly, there is one and only one point of arc $\overset{\frown}{db}$ at the boundary of $A(P(u_i))$. As arc $\overset{\frown}{ba}$ is arc of circle $C(v_x)$, that is, it is inside current region $A(P(u_i))$, then, it has no intersection point with the boundary of $A(P(u_i))$. So does arc $\overset{\frown}{cd}$ due to being out of circle S'. Hence circle $C(u_{i+1})$ has two and only two intersection points with the current region $A(P(u_i))$. The situation ② is equal to situation ① when there is only one intersection point.

■

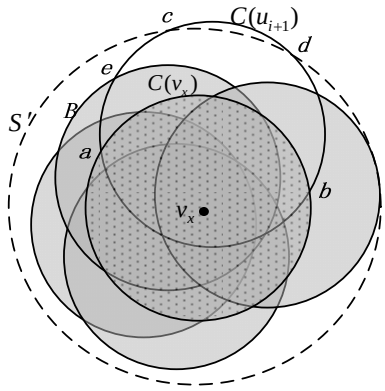

Figure 2. $C(u_{i+1})$ intersects with circle S'

Step 6) and step 7) of *algorithm*1 deal with the situation that there is no vertex in $Q(v_x)$ covered by region $A(u_{i+1})$. That is to find the nearest vertex u_j to u_{i+1} from $P(u_i)$ and calculate the intersection points of circle $C(u_j)$ and circle $C(u_{i+1})$, then add them into set $Q(v_x)$. So $Q(v_x)$ is updated. The correctness of this algorithm is ensured by the *theorem* 2 followed.

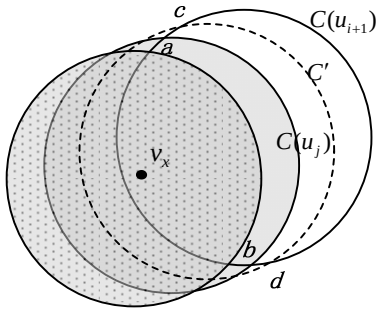

Figure 3. $C(u_{i+1})$ does not cover the intersection points of boundary

***Theorem* 2.** If $A(u_{i+1})$ does not cover any vertices in $Q(v_x)$ after adding circle $C(u_{i+1})$, and circle $C(u_j)$ is the nearest circle in $A(P(u_i))$ to circle $C(u_{i+1})$, then the two intersection points of circle and circle $C(u_{i+1})$ are at the boundary of $A(P(u_i))$.

***Proof* :**

As shown in Fig.3, the gray part represents $A(P(u_i))$. Prove by method of reduction to absurdity.

Assume circle is the nearest circle in $A(P(u_i))$ to circle $C(u_{i+1})$, and there is a circle C' so that its two intersection points with circle are at the boundary of $A(P(u_{i+1}))$. Circle $C(u_j)$ intersects circle $C(u_{i+1})$ at points a and b. Circle C' intersects circle $C(u_{i+1})$ at points c and d which both at the boundary of $A(P(u_i))$. Since only both point c and point d at the arc $\overset{\frown}{ab}$ of circle $C(u_{i+1})$ ensures that points c and d are located at the boundary of $A(P(u_i))$. Because circles $C(u_j)$ and C' are nearer to circle $C(v_x)$ than circle $C(u_{i+1})$, points a and b are covered by circle C'. So circle C' is nearer to circle $C(u_{i+1})$ than circle $C(u_j)$, which contradicts the assumption. Hence there is no circle C' like this.

■

Step 5) in *algorithm* 1 requires calculating updated set $Q(v_x)$ when $A(u_{i+1})$ covers some vertices in $Q(v_x)$, which is implemented by followed sub-algorithm.

We assume that points d_k, d_{k+1}, ... , d_{k+m} in $Q(v_x)$ are covered by $A(u_{i+1})$. As points d_k, d_{k+1}, ... , d_{k+m} are inside $C(u_{i+1})$, according to *theorem* 1 the arcs among points d_k, d_{k+1}, ... , d_{k+m} must be inside circle $C(u_{i+1})$ as well. So we can let d_k adjacent to d_{k+1}, d_{k+1} adjacent to d_{k+2}, … , d_{k+m-1} adjacent to d_{k+m} by arcs.

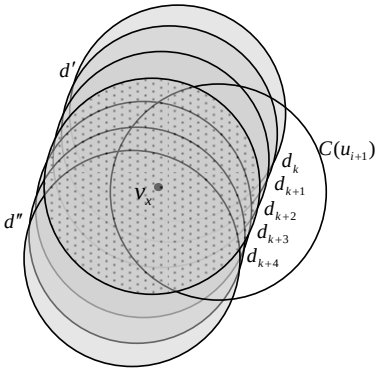

Figure 4. Added circle covers multi-intersection points of boundary

Let m=4, as shown in Fig.4 After adding circle $C(u_{i+1})$, $A(u_{i+1})$ covers points d_k , d_{k+1} , d_{k+2} , d_{k+3} , d_{k+4} in set $Q(v_x)$. If we use $b(\overset{\frown}{ab},\overset{\frown}{bc})$ to denote that point b is the intersection point of arc $\overset{\frown}{ab}$ and arc $\overset{\frown}{bc}$. If $\overset{\frown}{ab}$ is arc of circle $C(u)$ and $\overset{\frown}{bc}$ is arc of circle $C(v)$, then $b(\overset{\frown}{ab},\overset{\frown}{bc})$ can also be written as $b(u,v)$.

***Definition* 1.** b associates u and v, if b satisfies relationship $b(u,v)$.

Let arcs $\overset{\frown}{d'd_k}$, $\overset{\frown}{d_kd_{k+1}}$,, $\overset{\frown}{d_{k+2}d_{k+3}}$, $\overset{\frown}{d_{k+3}d_{k+4}}$, $\overset{\frown}{d_{k+4}d''}$ are arcs of circles respectively with u_d , u_k , u_{k+1} , u_{k+2} , u_{k+3} , u_{k+4} as the center. Points d_k , d_{k+1} , d_{k+2} , d_{k+3} , d_{k+4} can be

denoted by relationship $b(u,v)$ as: $d_k(u_d,u_k)$, $d_{k+1}(u_k,u_{k+1})$, $d_{k+2}(u_{k+1},u_{k+2})$, $d_{k+3}(u_{k+2},u_{k+3})$, $d_{k+4}(u_{k+3},u_{k+4})$. Considering that $d_k(u_d,u_k)$ and $d_{k+1}(u_k,u_{k+1})$ have a common vertex u_k, we define the new relationship of adjacent points.

***Definition* 2.** If adjacent intersection points and d_{k+1} have relationship $d_k(u_d,u_k)$ and $d_{k+1}(u_k,u_{k+1})$, thus the relationship of points d_k and d_{k+1} is defined as $\overline{d_k d_{k+1}}(u_d,u_{k+1})$, in which arc $\overline{d_k d_{k+1}}$ associates u_d, u_{k+1}.

According to the relationship among d_k, d_{k+1}, d_{k+2}, d_{k+3} and d_{k+4}, the adjacent points can be merged. For example, points d_k and d_{k+1} can be merged to be $\overline{d_k d_{k+1}}(u_d,u_{k+1})$ representing that arc $\overline{d_k d_{k+1}}$ associates u_d and u_{k+1}, then $\overline{d_k d_{k+1}}(u_d,u_{k+1})$ and $d_{k+2}(u_{k+1},u_{k+2})$ can be merged to be $\overline{d_k d_{k+2}}(u_d,u_{k+2})$. By the same method, d_k, d_{k+1}, d_{k+2}, d_{k+3} and d_{k+4} can be finally merged to be one formula $\overline{d_k d_{k+4}}(u_d,u_{k+4})$. If the whole $\overline{d_k d_{k+4}}$ is regarded as a point *b*, it can be derived that $A(u_{i+1})$ only covers one point *b*, and we get that $C(u_d)$ and $C(u_{k+4})$ are the right circles in current region whose intersection points with $C(u_{i+1})$ are at the boundary of region $A(u_{i+1})$.

Let set Q' to store covered points temporarily, set H for intermediate result. For convenience, let Q to denote $Q(v_x)$ in this algorithm.

Basing on the above analysis, the sub-algorithm to update set $Q(v_x)$ is as follows.

***Sub-algorithm*.** Update set $Q(v_x)$ in the case that boundary intersection points are covered

Input: set $Q(v_x)$, and vertex u_{i+1}.

Output: updated set $Q(v_x)$.

1) Initialize set $Q'=\varnothing$, $H=\varnothing$.

2) Pick an arbitrary unpicked point d_k from set *Q*, and calculate the distance d between d_k and u_{i+1}.

3) If $d \le R$, then put d_k into set Q'.

4) Return to step 2), until all points in *Q* are picked.

5) Let $Q = Q - Q'$.

6) Assuming the points in Q' to be $d_k, d_{k+1}, \ldots, d_{k+m}$, pick an arbitrary point d_{k+i} (i=0, 1, 2…) from Q', and put vertices u_{k+i} and v_{k+i} which associates with d_{k+i} into *H*.

7) Choose an unpicked point $d_{k+j}(j \ne i) = \{(x_{k+j}, y_{k+j}); (u_{k+j}, v_{k+j})\}$ from Q', such that $(u_{k+i}=u_{k+j}) \cup (u_{k+i}=v_{k+j}) \cup (v_{k+i}=u_{k+j}) \cup (v_{k+i}=v_{k+j})$ is true.

8) If $u_{k+i}=u_{k+j}$, then replace vertex u_{k+i} of *H* by v_{k+j}. If $u_{k+i}=v_{k+j}$, then replace vertex u_{k+i} of *H* by u_{k+j}. If $v_{k+i}=u_{k+j}$, then replace vertex v_{k+i} of *H* by v_{k+j}. If $v_{k+i}=v_{k+j}$, then replace vertex v_{k+i} of *H* by u_{k+j}.

9) Return to step 7), until there is no eligible point.

10) Let the vertices of *H* to be *u* and *v*, then calculate the points $a=\{(x_a,y_a);(u,u_{i+1})\}$ and $b=\{(x_b,y_b);(u,u_{i+1})\}$ intersected by $C(u)$ and $C(u_{i+1})$, and $c=\{(x_c,y_c);(v,u_{i+1})\}$ and $d=\{(x_d,y_d);(v,u_{i+1})\}$ by $C(v)$ and $C(u_{i+1})$.

11) If there is a point $p \in P(u_i)$ whose distance to certain one of *a*, *b*, *c*, *d* is less than *R*, then delete the certain point which may be any one of *a*, *b*, *c*, *d*.

12) Repeat step 11), until there are only two points left of *a*, *b*, *c*, *d*. Then put those two into set *Q*.

13) End.

B Solving Problem B

The forwarding set of the arbitrary vertex *v* in graph *G* can be gotten by *algorithm* 1. Before calculating CDS, we assume that *algorithm* 1 has been applied to each vertex in graph *G* and *B*(.) (. indicates arbitrary vertex in set *V*) has also been acquired already.

While calculating CDS, special packet Pa is broadcasted. Each node in the network sets its status bit ss for reducing the number of redundant dominating nodes and dominating bit cs for identifying dominating nodes. A node sets its ss to 0 if it has received packet Pa and sets its cs to 1 according to step 2) of *algorithm* 2. The steps of the *algorithm* 2 are following.

***Algorithm* 2:** Calculating CDS

Input: B(.).

Output: CDS.

1) Initialize each node's ss bit to 1 and cs bit to 0.

2) Select a sink node *s* in set *V*, and set ss bit of sink node *s* to 0.

3) Node *s* broadcasts packet Pa (including the information of $B(s)$ in head) to $N(s)$ and waits for receiving packet Pa (including status bit ss) from its neighbors. If Pa has been received, *s* sets its cs to 1. If time is out, go to step 7).

4) Each node in $N(s)$ checks its ss, and if ss equals to 0, then go to step 7).

5) Each node in $N(s)$ sets its ss to 0 and checks out if it belongs to $B(s)$. If not, then go to step 7).

6) The node in $N(s)$ takes the place of node *s*, and continue step 3), until all nodes in $N(s)$ have broadcast packet Pa already.

7) Put all nodes whoes cs equals to 1 in the connected dominating set.

8) End.

Fig.5 shows the topology of a network with 13 nodes, in which node 1 is selected to be sink node *s*. Black node indicates confirmed dominating node, and white node indicates non-dominating node. Nodes 1, 3, 4, 5, 9 and 11 are selected to be dominating nodes successively. (Considering channel delay, retransmission caused by packet loss and distance between nodes in real networks, the

connected dominating set in Fig.5 is just one of the possible results.)

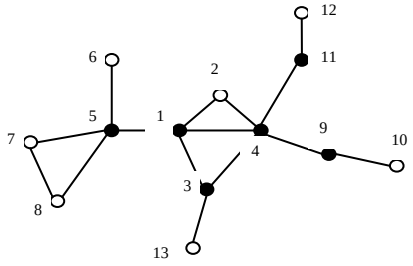

Figure 5. The connected dominating set of the network with 13 nodes

In *algorithm* 2, it effectively decreases the number of redundancy dominating nodes by setting ss. Node checks out if it will participate in transmission only by some simple calculating, and if it will, then it should set its cs to 1.

C Correctness Proof and Complexity Analysis

According to *algorithm* 2, if $G=(V,E)$ is a simple connected graph, then for arbitrary vertex v_x except the initial node, there must exist a v_y as the dominating node such that $v_x \in B(v_y)$. Hence the set (CDS) calculated by our algorithm is connected. Moreover, for an arbitrary vertex u_x in the set *V*, if u_x does not broadcast packet Pa, then there must be a vertex v_x which is confirmed to be dominating node and $u_x \in N(v_x)$ as shown in *algorithm* 1 and *algorithm* 2. Thus CDS calculated by algorithm OHCDS is the connected dominating set.

The complexity of *algorithm* 1 is mainly caused by following three parts: 1) Order the adjacent vertices of v_x by the distance to v_x. 2) Check out whether the circle $C(u_{i+1})$ added covers any boundary intersection points or not, if so, then go to *sub-algorithm*, 3) else choose a circle $C(u_j)$ nearest to circle $C(u_{i+1})$.

Suppose that Δ indicates the degree of the node. If we adopt QuickSort in the first part, the complexity of this part is $O(\Delta \log_2 \Delta)$. In the second part, while judging if the added circle covers any boundary intersection points or not, the average times needed to judge increases linearly. Thus its complexity for this part is $O(\Delta)$. As the situation that at least two boundary intersection points are covered by the added circle is dealt with in *sub-algorithm*, its time complexity is from $O(\Delta^2)$ as worst to $O(1)$ as best and $O(\Delta)$ as average. Step 11) and 12) of *sub-algorithm* are to choose vertices so that their complexity can be ignored. So the time complexity of the second task is $O(\Delta^2)$. On account of picking vertex u_j in $P(u_i)$, its comparing times increases linearly, so the time complexity of the third task is $O(\Delta)$.

Because nodes communicate with each other by broadcasting in *algorithm* 2, its time complexity can be ignored. So the time complexity of our algorithm OHCDS is $O(\Delta^2)$.

D Performance Bound Analysis

The covering problem is stated as follows: What is the minimum number of circles required to completely cover a given 2-dimensional space? It is known that no arrangement of circles could cover the plane more efficiently than the hexagons with sides *R* arrangement shown in Fig.6.

The best-case performance bound of OHCDS can be derived from covering problem. OHCDS gets approximate hexagonal vertices by calculating the minimum forwarding set and constructing a connected dominating set. It is assumed that the area of the network *A* is large compared to the area of one hexagon πR^2.The number of hexagons required to cover the entire network of area *A* is $\frac{A}{3\sqrt{3}R^2/2}$. The distance between two hexagon centers is greater than *R* and less than 2*R*, thus the best-case performance bound of OHCDS is $\frac{2A}{3\sqrt{3}R^2/2}$.

The worst-case performance bound is determined by the maximum number of transmissions. When this situation occurs, a node transmits only $R/2+\varepsilon$ distance, where ε is an infinitesimal. Thus the worst-case performance bound is $\frac{2A}{3\sqrt{3}(\frac{R}{2}+\varepsilon)^2/2}$. Ignoring ε, we can get the worst-case performance bound is $\frac{8A}{3\sqrt{3}(R)^2/2}$.

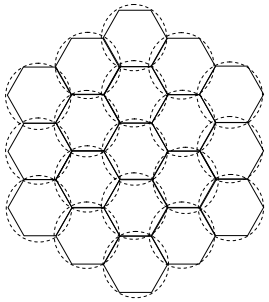

Figure 6. Covering a plane with circles in an efficient way Experimental Results

V. EXPERIMENTAL RESULTS

To test the performance of our algorithm (OHCDS), some simulation is done among our algorithm OHCDS, Dai's algorithm [13] and MISB algorithm [14] to compare their message number (communication traffic), convergence time and the size of CDS. Message number indicates how many broadcasting messages are needed by all of the nodes to accomplish the whole algorithm. Convergence time indicates how much time has been spent since the algorithm beginning to the end. Assume that time-unit means broadcasting interval between adjacent nodes and this broadcasting interval is same between any two adjacent nodes in the network. Channel delay and retransmis-

sion caused by packet loss and processing time of the node are ignored. We define convergence time as the number of broadcasting intervals since the algorithm beginning to the end. The topology of the network is formed according to the following principles:

- Nodes are randomly dispersed in a $1000m \times 1000m$ square area.
- The communication range of each node is 150m.
- The size of network is from 100 nodes to 450 nodes with increment of 50 nodes respectively.
- The simulation result is the average of 50 testing results with 8 different node groups.

In the experiments, we compare these three algorithms' performances on message number, convergence time and the size of CDS based on different node density and show results in Fig.7, 8, and 9. Experiments are as following.

Experiment 1 Compare the message number of our algorithm with those of Dai's algorithm and MISB. The result is as Fig.7 shows.

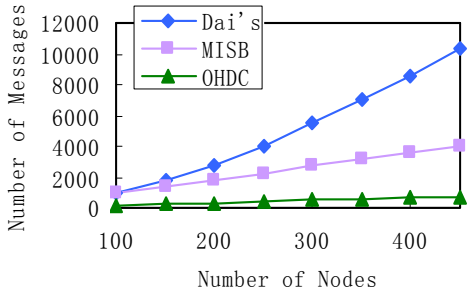

Figure 7. Comparison of the message number among OHCDS algorithm, Dai's algorithm and MISB algorithm

As OHCDS uses only 1-hop adjacent node information, the message number of OHCDS is mainly in the process of neighbor discovery and broadcasting in *algorithm* 2. The message number of neighbor discovery equals to the number of nodes, and the message number of broadcasting in *algorithm* 2 equals to the number of nodes that broadcast messages including the minimum forwarding set information. For this reason, the message number of our OHCDS algorithm is less than the double number of nodes in the network. As Fig.7 shows, the message number of Dai's algorithm is about 20 times the number of nodes, while the message number of MISB algorithm is about 9 times the number of nodes. So OHCDS is much better.

Experiment 2 Compare the convergence time of our algorithm with those of Dai's and MISB. To simplify the simulation, we assume that the neighbor discovery time in our algorithm is 1. The result is as Fig.8 shows.

Because *algorithm* 1 is executed severally, most of time is consumed by *algorithm* 2. Furthermore, it adopts broadcasting diffusion in *algorithm* 2, so time spent would not be more than the maximum route length. However, time spent on MISB algorithm is about 5 times the route length. Fig.8 depicts the comparison of the convergence time among these three algorithms with different node density. We can see that the convergence time of OHCDS approximately equals to that of Dai's algorithm with very low node density. However, with node number increasing in the network and node density increasing, the convergence time of Dai's algorithm increases rapidly, but the convergence time of OHCDS is almost invariant as the region is constant. We can also see in Fig.8 that the convergence time of OHCDS is one fifth of that of MISB.

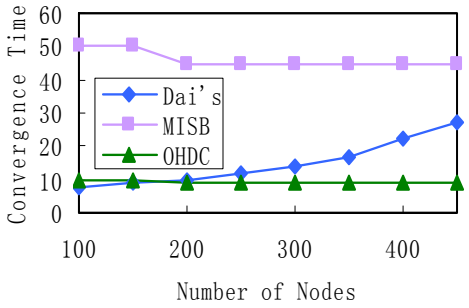

Figure 8. Comparison of the convergence time among OHCDS algorithm, Dai's algorithm and MISB algorithm

Experiment 3 Compare the CDS size of our algorithm with those of Dai's and MISB.

As shown in Fig.9, the CDS size of our algorithm is between that of Dai's algorithm and MISB algorithm. With node density increasing, the size of CDS of our algorithm is increasing obviously. It is mainly because of using communication coverage method in algorithm 1.

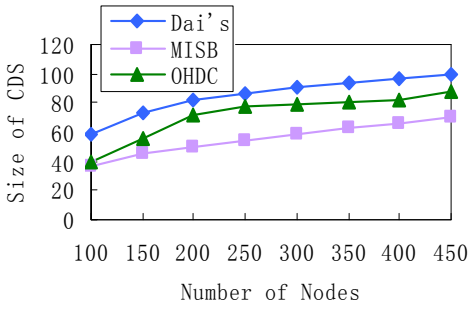

Figure 9. Comparison of the CDS size among OHCDS algorithm, Dai's algorithm and MISB algorithm

Summing up the above, we can see that our algorithm has the best performance on both message number and convergence time and also get small CDS size.

VI. Conclusion

Comparing to the other algorithms, there are a lot of advantages of our algorithm due to calculating connected dominating set using only 1-hop neighbor information. As simulation results show, less communication cost and less convergence time is achieved by our algorithm. Less message number means less energy consumption, and that less convergence time indicates faster calculating. Our algorithm can also get a smaller CDS size with lower node density. Meanwhile, our algorithm is low energy consumption and low convergence time, and it is applicable to dynamic wireless sensor networks. As for static network, our algorithm achieves energy balance. The future work will focus on optimizing the CDS size on the assumption of keeping its present advantages.

ACKNOWLEDGMENT

This paper is supported by Research Fund for the Doctoral Program of Higher Education of China for New Teachers No.200806141110 and project CNGI of NDR No.CNGI-09-01-07.

REFERENCES

[1] S.C.H. Huang, M. Sun, P. Wan, X. Jia, "Interference-Aware, Fully-Distributed Virtual Backbone Construction and its Application in Multi-Hop Wireless Networks", IEEE Transactions on Communications, vol.58, no.12, pp. 3550-3560, 2010.

[2] Dai F, Wu J, "Performance analysis of broadcast protocols in adhoc networks based on self-pruning", IEEE Trans on Parallel and Distributed Systems, vol. 15,no.11, pp. 1-13, 2004.

[3] Lee J, Mans B, "Energy-efficient virtual backbones for reception-aware MANET", Proceedings of the 63rd IEEE Vehicular Technology Conference, Melbourne: IEEE Press, 2006: 1097-1101.

[4] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and Ker-I Ko, "A greedy approximation for minimum connected dominating sets", Theoretical Computer Science, vol. 329, issues1–3, pp. 325–330, Dec. 2004.

[5] S. Butenko, X. Cheng, C. Oliveira, P. M. Pardalos, "A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks", Recent Developments in Cooperative Control and Optimization, pp. 61-73, New York:Kluwer Academic Publishers, 2004.

[6] M. Min, H. Du, X. Jia, C. Huang, S. Huang, and W. Wu, "Improving construction for connected dominating set with Steiner tree in wireless sensor networks", Journal of Global Optimization, vol.35, no.1, pp.111-119, 2006.

[7] Lu G, Zhou MT, Tang Y, Zhao MY, Niu XZ, and She K, "Approximation algorithms for the connected dominating set problem in unit disk graphs", Journal of Electronic Science and Technology of China, vol.3, no.7, pp. 214-222, 2009.

[8] A. Vahdatpour, F. Dabiri, M. Moazeni, and M. Sarrafzadeh, "Theoretical Bound and Practical Analysis of Connected Dominating Set in Ad Hoc and Sensor Networks", Distributed Computing, vol.5218, pp.481-495, 2008.

[9] Joseph P. Macker, "Distributed connected dominating set election from uniform random to power law network graphs", Military Communications Conference, pp.1-7, 2009.

[10] Wu J, Lou W, Dai F, "Extended multipoint relays to determine connected dominating sets in MANETs", IEEE Transactions on Computers, vol.55, no.3, pp.334-347, 2006.

[11] H. Du, Q. Ye, J. Zhong, Y. Wang, W. Lee and H. Park, "PTAS for Minimum Connected Dominating Set with Routing Cost Constraint in Wireless Sensor Networks", Combinatorial Optimization and Applications, vol.6508, pp.252-259, 2010.

[12] Dai F, Wu J, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks", IEEE Transactions on Parallel and Distributed Systems, vol.15, no.10, pp.905-920, 2004.

[13] Tang Y, Zhou M, "Maximal Independent Set Based Distributed Algorithm for Minimum Connected Dominating Set", Acta Electronica Sinica, vol.35, no.5, pp.868-874, 2007.(in Chinese)

[14] Sakai K., Sun M, Ku W. "Fast Connected Dominating Set Construction in Mobile Ad Hoc Networks", IEEE International Conference on Communications (ICC'09), Germany: IEEE Press, 2009: 1-6.

[15] Wenyong Wang, Jun Zhang, Yong Tang, Yu Xiang, Ting Yang, "One-Hop Neighbor Transmission Coverage Information Based Distributed Algorithm for Connected Dominating Set", 3rd International Symposium on Parallel Architectures, Algorithms and Programming, pp.15-21, December 2010.

Wang Wenyong was born in Sichuan, China, in 1967. He is a professor of University of Electronic Science and Technology of China. He received his Ph.D. from University of Electronic Science of China. He is a member of China Education and Research Network Committee of Experts, senior member of China Computer Federation, member of Internet Professional Committee, member of Software Engineering Professional Committee, and member of High Performance Computing Professional Committee. His research interest lies in wireless sensor network and network computing.

Zhang Jun was born in Gansu, China, in 1985. He is a engineer of Huawei Technologies Company. He received his M.S. from University of Electronic Science and Technology of China in 2011.

Tang Yong was born in Yunnan, China, in 1973. He is an associate professor of University of Electronic Science and Technology of China. He received his Ph.D. from University of Electronic Science and Technology of China in 2007, M.S. from Xidian University in 1999, and B.S. from Xidian University in 1996. Prior to reading his doctorate in computer science, he served as a senior engineer for DSET (China), working on the projects of TMN. His research interest lies in wireless sensor network and network management.

Xiang Yu was born in Sichuan, China, in 1973. He is an associate professor of University of Electronic Science and Technology of China. He received his B.S., M.S., Ph.D. from University of Electronic Science of China. His research focuses upon wireless sensor network and wireless communication.

Yang Ting was born in Sichuan, China, in 1975. He is a lecturer of University of Electronic Science and Technology of China. He received his M.S. from University of Electronic Science of China. He is currently a Ph.D. candidate in Computer Science in University of Electronic Science and Technology of China. His research fields include wireless sensor network and network security.