This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

A New Constant Factor Approximation for Computing 3-Connected m-Dominating Sets in Homogeneous Wireless Networks

Donghyun Kim*, Wei Wang†, Xianyue Li‡, Zhao Zhang§, and Weili Wu*
*Department of Computer Science, The University of Texas at Dallas, Richardson, Texas 75080, USA
E-mail: donghyunkim@student.utdallas.edu, weiliwu@utdallas.edu
† Department of Mathematics, Xi'an Jiaotong University, Xi'an, P.R. of China, 710049
E-mail: wang_weiw@163.com
‡School of Mathematics and Statistics, Lanzhou University, Lanzhou, Gansu, P.R. of China, 730000
E-mail: lixianyue@lzu.edu.cn
§College of Mathematics and System Sciences, Xinjiang University, Urumqi, Xinjiang, P.R. of China, 830046
E-mail: hxhzz@163.com

***Abstract*—In this paper, we study the problem of constructing quality fault-tolerant Connected Dominating Sets (CDSs) in homogeneous wireless networks, which can be defined as minimum k-Connected m-Dominating Set ((k, m)-CDS) problem in Unit Disk Graphs (UDGs). We found that every existing approximation algorithm for this problem is incomplete for $k \geq 3$ in a sense that it does not generate a feasible solution in some UDGs. Based on these observations, we propose a new polynomial time approximation algorithm for computing $(3, m)$-CDSs. We also show that our algorithm is correct and its approximation ratio is a constant.**

I. Introduction

Over many years, wireless networks such as ad-hoc or sensor networks have emerged as popular research topics in the networking community. Typically, wireless networks consist of a number of wireless nodes, which communicate with each other using Radio Frequency (RF) signals. They are convenient to deploy and can construct instant networks in most application fields without any pre-existing infrastructure. In addition, their cost is getting lower due to the continuous improvements in related technologies. So far, numerous applications of wireless networks, such as conferences, traffic control, battlefield surveillance, environmental monitoring, disaster rescue, concert, health applications, etc., are identified and developed [1], [2].

In wireless networks, each node has a limited signal transmission range. A node can directly send a message to another node if they are closer than the sender's maximum communication range. Otherwise, the message has to be relayed through multiple intermediate nodes. Generally, energy-efficiency is a crucial issue of wireless networks since each node has a limited energy source such as a battery. It is known that wireless networks consume most energy for communications rather than the other activities. In wired networks, finding and maintaining routing paths can be performed efficiently with the help of some pre-existing infrastructure (i.e. a backbone network). On the contrary, in wireless networks, most routing protocols rely on a flooding-like scheme in which every node participates in the route discovery by broadcasting control messages it receives. Unfortunately, it is known that such flooding scheme causes huge amount of collision and redundancy and thus is very energy exhausting [3]. Therefore, to extend the lifetime of wireless networks and make them more useful, it is essential to have an efficient routing strategy.

A Virtual Backbone (VB) of a wireless network is a subset of nodes such that 1) every node in a given network is either in the subset or adjacent to a node in the subset and 2) a subgraph induced by the subset is connected. Given a VB, any two nodes can communicate with each other through the backbone structure. Therefore, once a VB is defined in a wireless network, any conventional routing algorithm for the wireless network can be used over the backbone structure. Employing a VB into a wireless network has several apparent benefits. First of all, since the number of nodes involved in message routing is reduced, the amount of wireless signal collision and interference will be diminished. This approach also makes a routing path search space smaller, and thus any routing protocol can converge faster. Therefore, any routing protocol running over a VB becomes more efficient [4].

Ever since the idea of employing VB for wireless networks is introduced [5], a huge amount of effort has been made to find a better quality VB. Clearly, the benefits of a VB can be magnified by making its size (the number of nodes constituting the VB) smaller. Guha and Kuller first modeled the problem of computing the smallest VB as the minimum Connected Dominating Set (CDS) problem in general graphs [6], which is a well-known NP-Hard problem [7]. In addition, they introduced two approximation algorithms having a worst case performance guarantee. Nowadays, this approach becomes one of major ways to compute quality CDSs with a performance guarantee.

In some wireless networks such as ad-hoc networks, nodes are mobile and thus the topology of such networks can be

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

changed frequently. In the mobile wireless networks, a VB induced by a CDS can be broken easily and thus it should be re-computed repeatedly [8]. To make a VB more resilient in volatile wireless networks, the fault-tolerance of the VB is considered. In [9], a k-connected m-dominating set is introduced as a generalized abstraction of a fault tolerant VB, which satisfies following properties: Let $G = (V, E)$ be a graph representing a wireless network and C be a VB. Then, first C has to be k-connected so that the VB can survive after any $k-1$ backbone nodes fail. In addition, a node $x \in (V-C)$ has to adjacent to at least m nodes in C so that x can be connected even after $m-1$ adjacent backbone nodes fail. Here, both k and m are two system parameters determining the degree of fault-tolerance.

In this paper, we study the problem of computing k-connected m-dominating sets in Unit Disk Graphs (UDGs), which represent homogeneous wireless networks. We found that so far several approximation algorithms have been proposed to compute k-connected m-dominating sets for any $k \geq 3$ and a positive constant m, but none of them is perfect in a sense that each of them does not produce a feasible solution in some UDGs. Based on those observations, we propose a new constant factor approximation algorithm for computing 3-connected m-dominating sets in UDGs given a constant m. Our algorithm first computes a 2-connected m-dominating set using an existing constant factor approximation [10], [11]. Then, it augments the subset into a 3-connected m-dominating set. We prove that the algorithm produces a feasible solution in any 3-connected UDG and has a constant factor Performance Ratio (PR) given a constant m.

The rest of this paper is organized as follows. Section II includes several important notations, definitions, lemmas, and theorems. In Section III, we introduce related work. In Section IV, we present a new constant factor approximation algorithm for computing 3-connected m-dominating sets in UDGs with its performance, correctness, and running time analysis. Finally, Section V concludes this paper.

II. Notations and Definitions

Now, we introduce several important notations and definitions. $G = (V, E) = (V(G), E(G))$ is a graph. For any $u, v \in V(G)$, $udist(u, v)$ is the Euclidean distance between them. We shall use $G - S$ to denote the induced subgraph on the vertex set $V(G) - S$ for $S \subset V(G)$.

Definition 2.1: G is a ***Unit Disk Graph (UDG)*** if $\forall u, v \in V(G)$, there is a bidirectional edge between them if and only if $udist(u, v) \leq 1$.

Huson and Sen suggested to use UDG to abstract homogeneous wireless networks for the first time [12]. Nowadays, this approach is widely used in many literatures. More general properties of UDG can be found in [13].

Definition 2.2: $D(G) \subseteq V(G)$ is a ***Dominating Set (DS)*** of G if $\forall v \in V(G)$, either $v \in D(G)$ or $\exists u \in D(G)$ such that $(v, u) \in E(G)$.

Definition 2.3: An ***Independent Set (IS)*** of G is a subset $I(G) \subset V(G)$ such that $\forall u, v \in I(G)$, $(u, v) \notin E(G)$.

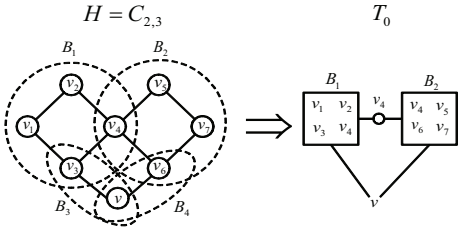

Fig. 1. The right figure is the block graph T_0 of $H - \{v\}$. Each maximal 2-connected subgraph in $H\{v\}$ becomes a block in T_0 and each bad-point of H in $H - \{v\}$ becomes a cut-vertex in T_0. Suppose a G_3 is given and H is a subgraph containing a $C_{2,3}$ of G_3. Since v_4 is a cut-vertex of T_0 (which means that $\{v, v_4\}$ is a separator of H), there has to be a H-path from B_1 to B_2 in T_0. Then, in the left graph, v_4 cannot be any of the ends of such H-path.

Definition 2.4: An IS I is a ***Maximal Independent Set (MIS)*** if any $v \in (G(V) - I)$, $I \cup \{v\}$ is not an IS anymore.

Since an IS is a kind of DS and computing a minimum DS is NP-hard [21], finding a minimum IS is also NP-hard.

Definition 2.5: A subset $C(G) \subseteq V(G)$ is a ***Connected Dominating Set (CDS)*** of G if 1) $C(G)$ is a DS of G and 2) the graph induced by $C(G)$ is connected.

One typical way to compute a CDS is 1) computing an MIS using a coloring algorithm and 2) selecting an additional set of nodes to connect the MIS using an algorithm for Steiner minimum tree problem or minimum spanning tree problem. Guha and Kuller first used the minimum CDS problem for computing a minimum size VB for wireless networks [6], and later this becomes a popular approach in this research field.

Definition 2.6: A DS $D(G) \subseteq V(G)$ is a ***m-Dominating Set (m-DS)*** of G if $\forall v \in V(G) - D(G)$, v is adjacent to at least m nodes in $D(G)$.

Definition 2.7: A graph G is ***k-vertex-connected*** if G is still connected after any $k-1$ nodes are removed from G.

In the rest of this paper, "k-connected" and "k-vertex-connected" will be used interchangeably. Also, G_k is a k-connected graph.

Definition 2.8: A CDS $C(G) \subseteq V(G)$ is ***k-Connected m-Dominating Set*** if 1) the graph induced by $C(G)$ is k-connected and 2) $C(G)$ is a m-DS.

In this paper, (k, m)-CDS and $C_{k,m}$ are mutually used to denote a k-connected m-dominating set. Also, k-CDS is a (k, k)-CDS. Finally, $C^{opt}_{k,m}$ is an optimal $C_{k,m}$ in a graph G.

Definition 2.9: $v \in V(G)$ is called a ***cut-vertex*** if the induced graph of $G - \{v\}$ is disconnected.

Definition 2.10: A ***block*** is a maximal connected subgraph of G including no cut-vertex.

Note that every block in a connected graph is either a maximal 2-connected subgraph or a bridge (an edge with two end points).

Definition 2.11 ([14]): Given a connected graph G consisting of blocks and cut-vertices, we construct a ***block graph*** G' as follow: For each block B in G, we have a corresponding node v_B in G'. For each cut-vertex u in G, we have a node u in G'. Finally, there is an edge between v_B and u in G' if

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

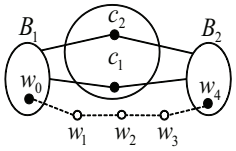

Fig. 2. Given a G_3, a $C_{2,3} \subset G_3$, and a separator (c_1, c_2) of $C_{2,3}$, suppose $C_{2,3} - \{c_1, c_2\}$ is divided into several connected components, $\{B_1, B_2, \cdots\}$. Then, there have to be two different components B_i and B_j such that the length of the shortest H-path between them is at most three hops.

$u \in B$ in G. Figure 1 shows one example of constructing a block graph.

In this paper, we say a block graph of a block consists of **sub-blocks**. Observe that 1) two different blocks of G share at most one cut-vertex in common, 2) every edge of G lies in a unique block, and 3) G is the union of its blocks. Then, Proposition 2.1 naturally follows based on those observations.

Proposition 2.1 ([14]): The block graph of a connected graph is a tree.

Definition 2.12: For a connected graph G, we call a pair of vertices $\{u, v\} \subseteq V(G)$ a ***separator*** of G if the subgraph induced by $G - \{u, v\}$ is disconnected.

Definition 2.13: For a G_2, $v \in V(G_2)$ is a ***good-point*** if the subgraph induced by $G_2 - \{v\}$ is still 2-connected. Otherwise, it is a ***bad-point***.

Definition 2.14: An ***H-path*** of a subgraph H of a graph G is a path connecting two nodes in H such that no internal node of the path is in H.

In the rest of this paper, H_k-path means an H-path whose length is at most k.

Now, we introduce several important lemmas and theorems.

Lemma 2.2: A 2-connected graph without any bad-point is 3-connected.

Proof: Note that by definition, a graph G is 3-connected if to disconnect G, we need to remove at least three nodes. Suppose G_2 is a 2-connected graph having only good points. Then, when a node v is removed from G_2, the graph should be still 2-connected. Next, to make $G_2 - \{v\}$ disconnected, we still need to delete at least two other nodes. Therefore, this lemma is true. ■

Lemma 2.3 ([10]): Given a G_3, a $C_{2,3} \subset G_3$, and a separator $\{c_1, c_2\}$ of $C_{2,3}$, suppose $C_{2,3} - \{c_1, c_2\}$ is divided into several connected components, $\{B_1, B_2, \cdots\}$. Then, there have to be two different components C_i and C_j such that the length of the shortest H-path between them is at most three hops.

Proof: Suppose this lemma is not true and the distance between any two closest components B_1 and B_2 is at least four hops (See Figure 2). Now, suppose the shortest path between them is $\{w_0, w_1, w_2, \cdots, w_l\}$. Then, at least one node in $\{w_2, \cdots, w_{l-1}\}$ has to be connected to other components since at best it can be dominated by c_1 and c_2 only. This contradicts to the assumption that B_1 and B_2 are two closest components, and thus $l \leq 3$ has to be true. ■

Lemma 2.4: Given a G_3, a $C_{2,3} \subset G_3$, and a separator $\{c_1, c_2\}$ of $C_{2,3}$, $C_{2,3} - \{c_1, c_2\}$ can be divided into at most five components.

Proof: In UDGs, a node can have at most five independent neighbors [19]. Since the components have to be independent and connected to c_1 (and c_2), their number cannot be more than five. ■

*Theorem 2.5 (**Menger's Theorem** [22]):* Given a graph G and two vertices $u, v \in V(G)$, the minimum number of vertices separating u from v in G is equal to the maximum number of disjoint $u - v$ paths in G.

III. Related Work

The necessity of (k, m)-CDS is initially discussed by the seminary work of Dai and Wu [15]. They introduced three localized algorithms to construct k-CDSs in general graphs. However, no theoretical analysis is given for their worst case performance. Since computing an optimal (k, m)-CDS is NP-hard even in the simplest case, where $k = m = 1$, many efforts were made to introduce an approximation algorithm, which has a worst case performance guarantee on the size of resulting (k, m)-CDSs [9]–[11], [16]–[18].

Wang et al. proposed an approximation algorithm, CDS Augmentation Algorithm (CDSA), to compute (k, m)-CDSs in UDGs, where $k = 2$ and $m = 1$ [9]. CDSA first computes 1-CDS, noted by $C_{1,1}$, using any existing approximation algorithm. Then, it finds a maximal 2-connected subgraph of the $C_{1,1}$ and augments this subgraph by adding paths iteratively. They also proved that the PR of their algorithm is 62.19. Shang et al. [10] and Thai et al. [11] independently introduced approximation algorithms to compute $(1, m)$-CDSs in UDGs and DGBs, respectively. Their schemes first compute a $C_{1,1}$ and iteratively find $m-1$ MISs for the remaining nodes and output the union of $C_{1,1}$ and $m - 1$ MISs. Clearly, after the algorithms are finished, for any node not in the output must have at least m neighbors in the output. At the end, they showed the approximation ratio of their algorithms is $O(1)$, given a constant m. More formal proof and approximation analysis can be found in [10] and [11].

Thai et al. [11] first introduced a centralized approximation algorithm to compute (k, m)-CDSs in DGBs in addition to approximation algorithms to compute $(1, m)$-CDSs and (k, k)-CDSs. Later, based on this result, Zhang et al. [16] built an approximation algorithm which has approximation ratios for both size and diameter of resulting (k, m)-CDSs. The first distributed approximation algorithm for this problem is introduced by Wu et al. [17]. More recently, Wu et al. [18] made both centralized and distributed approximation algorithms for this problem. In Appendix, we explain why the proposed approximation algorithms to compute (k, m)-CDSs for any $k \geq 3$ and $m \geq 1$ in [11], [16]–[18] are flawed.

IV. A Constant Factor Approximation Algorithm for Computing 3-Connected m-Dominating Sets in Unit Disk Graphs

In this section, we introduce our algorithm, Fault-Tolerant Connected Dominating Sets Computation Algorithm (FT-CDS-CA), which computes $(3, m)$-CDSs in UDGs. We first

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

proceed this section with $m = 3$ and later explain how our result can be generalized for any $m \neq 3$. Roughly, FT-CDS-CA works as follows: Given a G_3, FT-CDS-CA first computes a $C_{2,3} \subseteq G_3$ using a constant factor approximation algorithm in [10], [11] and set $Y \leftarrow C_{2,3}$. Next, the algorithm identifies the set of all bad-points X in $C_{2,3}$. Naturally, $|X| \leq |C_{2,3}|$. By Lemma 2.2, the 2-connected subgraph $C_{2,3}$ is 3-connected if $X = \emptyset$. The core strategy of FT-CDS-CA is that the algorithm repeatedly changes each bad-point $v \in X$ into a good point by moving at most some constant number of nodes from $G_3 - Y$ to Y without introducing any new bad-point into X. In this way, after changing all bad pints in X into good, Y becomes 3-connected 3-dominating set, and accordingly, the total number of nodes newly added to $C_{2,3}$ is upper bounded by a constant factor of $|C_{2,3}|$.

A. How to Convert a Bad-Point to a Good-Point?

FT-CDS-CA is a round-based algorithm. In each round, it converts one bad-point in Y to a good-point as follows: Given a 2-connected subgraph Y (initially, this is a $C_{2,3}$) and the set X of bad-points in Y, we select $v \in X$ and compute a leaf-block tree T_0 of $Y - \{v\}$. Then, T_0 constitutes of a set of blocks $\{B_1, B_2, \cdots, B_s\}$ and a set of cut-vertices $\{c_1, \cdots, c_t\}$. By Lemma 4.1, v can constitute a separator only with another node in $\{c_1, \cdots, c_t\}$. Suppose T_0 has no $w \in B_i$ for all $1 \leq i \leq s$ such that $w \notin \{c_1, \cdots, c_t\}$ and $\exists j, \{w, c_j\}$ is a separator of Y. Then, by Lemma 4.2 and Lemma 4.3, FT-CDS-CA makes one c_k for $1 \leq k \leq t$ to be a good-point by adding constant number of H_3-paths such that $Y - \{c_k\}$ is still 2-connected (Line 21-34 of Algorithm 1). Otherwise, we have B_i having a separator $\{w, c_j\}$ of Y in it. Again, by Lemma 4.1, any node v' in $B_i \cap (X - \{c_j\})$ can constitute a separator only with another node in $B_i \cap (X - \{v'\})$. Note that by switching c_j to v, we can make the original problem smaller. By repeating such process, we can keep making our problem smaller and eventually can find a block B_i in which our strategy works.

Now, we introduce the detailed description of our algorithm for the conversion, which consists of following two discrete steps, 1) Multi-Level Decomposition, and 2) One Bad-Point Elimination. Note that the first step takes a polynomial time since in each level, we can compute a leaf-block tree within a polynomial time and at most a polynomial number of trees have to be computed. Also, the second step takes a polynomial time to make one bad-point to a good-point. Since at most a polynomial number of rounds are repeated to eliminate all bad-points and find a subgraph $C_{3,3}$, the algorithm is clearly polynomial time executable.

1) Multi-Level Decomposition: The purpose of this step is for finding a subgraph in which the second step can be applied to convert at least one bad-point in X into a good point. We assume that $X \neq \emptyset$, since otherwise Y is already 3-connected. Given a 2-connected block $B \leftarrow Y$, FT-CDS-CA first picks $v \in X$ and starts the initial decomposition process (say level-0 decomposition). Then, $B - \{v\}$ is decomposed into a (level-0) block graph T_0, which is a tree whose vertices consist of a set of sub-blocks $\{B_1, \cdots, B_s\}$ and a set of cut-vertices $\{c_1, \cdots, c_t\}$. Observe that after the level-0 decomposition, $s \geq 2$ and $t \geq 1$.

Now, FT-CDS-CA examines each sub-block B_i in T_0 to see if there is a pair of nodes (i.e., an internal node w of B_i and a cut-vertex c_j of T_0), which can constitute a separator of B. Note that such w and c_j have to be in the same 2-connected subgraph of B, while this is not true in T_0 (they are shown separately in T_0). We will call such sub-block a ***bad sub-block***. Once we find a bad sub-block B_i, we stop searching, set $B \leftarrow B_i$, $v \leftarrow c_j$, $u \leftarrow w$, and start next level (level-1) decomposition process on B. If such pair does not exist, we can start the second step, "one bad-point elimination" on B.

Generally, after the level-l decomposition of $B - \{v\}$ for $l > 0$, the T_l may consist of one sub-block or more, while after the initial decomposition, T_0 must have at least two sub-blocks given v is a bad point. Now, we consider following two cases.

Case 1: T_l consists of one sub-block. In this case, $T_l = B_1$ itself. We first try to find $w \in B_1$ such that $\{u, w\}$ is a separator of Y. If so, we set $B \leftarrow B_1$, $v \leftarrow u$, $u \leftarrow w$, and start level-$(l+1)$ decomposition process. If such w does not exist, we can start the second step, "one bad-point elimination".

Case 2: T_l consists of at least two sub-blocks. In this case, we first search any bad sub-block in T_l. If exists, set $B \leftarrow B_i$, $v \leftarrow c_j$, $u \leftarrow w$, and start next level (level-$(l+1)$) decomposition process. If such pair does not exist, we can start the second step, "one bad-point elimination".

2) One Bad-Point Elimination: At this point, we have a leaf-block tree T_l, which is a leaf-block tree of $B - v$. Also, T_l includes no bad-sub-block and a bad-point v. If $s = 1$, we find an H_3-path H' from $B_1 - \{v, u\}$ to $Y - B_1$ and add H' to Y. After this process, u is not a bad-point anymore and thus it can be deleted from X. Then, this iteration is finished. If $s \geq 2$, we need to employ a simple process to make one of the cut-vertices connecting $\{B_1, B_2, \cdots, B_s\}$ to be a good point. Now, we explain our core strategy after introducing one definition which will make our writing more concise.

Definition 4.1: Given a leaf-block tree T_l which can be obtained after level-l decomposition of the original graph H, a ***leaf-block path*** is the path starting from a leaf block of T_l to a block or cut-vertex of T_l whose degree in H is greater than two.

First, we pick a leaf-block path

$$P = \{\tilde{B}_0, \tilde{c}_1, \tilde{B}_1, \cdots, \tilde{c}_i, \tilde{B}_i, \cdots\},$$

where $\tilde{B}_0, \tilde{c}_i$, and $\tilde{B}_i$ are a leaf sub-block, a cut-vertex, and an internal sub-block of T_l, respectively, such that in Y there is no path from any $\tilde{B}_i$ to $Y - B$ for $i \leq k$, where k is the number of cut-vertices in P. Here, note that P can end at some $\tilde{B}_k$ or $\tilde{c}_k$. We also denote $P_i = P - \{\tilde{B}_0, \cdots, \tilde{c}_i\}$. Next, we set i from 1 to $k-2$ and repeatedly check whether $\tilde{c}_i$ can become a good point by following ways.

Case 1: if there is an H_3-path H' from $\tilde{B}_i$ to $\tilde{B}_{i+1}$, then set $Y \leftarrow H \cup H'$ and finish this iteration, since $\tilde{c}_i$ is not a bad-point in Y anymore.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Case 2: if there are an H_3-path H' from $\tilde{B}_i$ to $(Y-P)\cup P_{i+1}$ and another H_3 path H'' from $\tilde{B}_{i+1}$ to $P-P_i$, then set $Y \leftarrow Y \cup H' \cup H''$ and finish this iteration, since $\tilde{c}_{i+1}$ is not a bad point in Y anymore.

If we cannot eliminate any of c_i for $1 \le i \le k-2$ using the sequential strategy above, we have to be in one of following two situations.

Case 1: if P ends at $\tilde{B}_k$, then there has to an H_3-path H' from $\tilde{B}_{k-1}$ to $Y-(P-\tilde{B}_k)$. Find such H' and set $Y \leftarrow Y \cup H'$. Since $\tilde{c}_{k-1}$ became a good point, this should be deleted from X.

Case 2: if P ends at $\tilde{c}_k$, then $Y-\{\tilde{c}_k, v\}$ can be partitioned into at most five parts. Then, we have to find four H_3-paths and add them to Y. Since $\tilde{c}_k$ became a good point, this should be deleted from X.

Algorithm 1 is the formal description of our algorithm. In the subsequent sections, we show the correctness proof, performance analysis, and time complexity of this algorithm.

B. Correctness and Performance Analysis

In this section, we prove that given any G_3 containing a solution, FT-CDS-CA can correctly generate a $C_{3,3}$. In addition, we show that the PR of our algorithm is a constant and the running time of our algorithm is polynomial.

Lemma 4.1: Let G_3 be a 3-connected graph and H be a 2-connected subgraph of G_3. Let v be a bad-point of H and T be a leaf-block tree which can be obtained after a decomposition of $H-\{v\}$. Suppose $V(T)=\{B_1,\cdots,B_s\}\cup\{c_1,\cdots,c_t\}$. Let $u \in B_i$ and $u \notin \{c_1,\cdots,c_t\}$. Then $\{u,w\}$ is not a separator of graph H for $w \in H-B_i$.

Proof: The assertion that $\{u,v\}$ is not a separator of H follows easily from the fact that $H-\{v\}$ is 1-connected and B_i is 2-connected. Thus, the deletion of u keeps the connectedness of $H-\{v\}$. Next, let $w \neq v$. We show $\{u,w\}$ is not a separator of H, i.e., $H-\{u,w\}$ is connected. Let $x,y \in V(H)-\{v,u,w\}$ be two distinct vertices, we shall show x and y can connect to each other in $H-\{u,w\}$. We distinguish two cases.

Case 1: $w \in B_j$ $(j \neq i)$ and $w \notin \{c_1,\cdots,c_t\}$. If x and y are contained in the same block B_k, then x and y can be connected to each other in $H-\{v,u,w\}$. This follows immediately when $k \neq i,j$; while if $k=i$ or $k=j$, since B_k is 2-connected, x and y can still be connected to each other after the deletion of u and w. Next, suppose that x and y are not contained in the same block, and let $(\tilde{B}_1,\tilde{c}_1,\tilde{B}_2,\tilde{c}_2,\cdots,\tilde{B}_q)$ $(q \ge 2)$ be a path in the leaf-block tree T connecting $\tilde{B}_1$ and $\tilde{B}_q$ such that $x \in \tilde{B}_1, y \in \tilde{B}_q$. Clearly, there is a path $P=(xP_1\tilde{c}_1P_2\tilde{c}_2\cdots\tilde{c}_{q-1}P_qy)$ in $H-\{v,u,w\}$ connecting x and y, where P_1 (resp. P_q) is a path contained in $\tilde{B}_1$ (resp. $\tilde{B}_q$) connecting x (resp. y) and c_1 (resp. c_{q-1}), and P_m $(2 \le m \le q-1)$ is a path contained in $\tilde{B}_i$ connecting $\tilde{c}_{m-1}$ and $\tilde{c}_m$. If u and w is not contained in any of $\tilde{B}_m$, then our assertion clearly holds. Now, suppose that u is contained in, say, $\tilde{B}_{m'}$ $(=B_i)$. Since $\tilde{B}_{m'}$ is 2-connected, there is a path $\tilde{Q}_{m'}$ (which is independent of $\tilde{P}_{m'}$) in $\tilde{B}_{m'}$ connecting $\tilde{c}_{m'-1}$ and $\tilde{c}_{m'}$. If w is also contained in

Algorithm 1 FT-CDS-CA (G_3)

1: Compute a $C_{2,3}$ and set $Y \leftarrow C_{2,3}$.
2: Identify the set of bad points X in Y.
3: **while** $X \neq \emptyset$ **do**
4: Set $Y \leftarrow$ SUB $(G_3, X, Y, Y, v, \cdot)$ for any $v \in X$.
5: **end while**
6: Return Y.
7: **function** (SUB (G_3, X, Y, B, v, u))
8: Construct a leaf-block tree T of $B-\{v\}$.
9: Note that $V(T)=\{B_1,\cdots,B_s\}\cup\{c_1,\cdots,c_t\}$.
10: **if** $s=1$ **then**
11: **if** there is $w \in B_1$ such that (u,w) is a separator of Y **then** $Y \leftarrow$ SUB $(G_3, X\cap B_1, Y, B_1, u, w)$.
12: **end if**
13: Find an H_3-path H' from $B-\{v,u\}$ to $Y-B$ and return $Y \leftarrow Y \cup H'$.
14: **else**
15: **for** $i=1$ to s **do**
16: **while** there is a separator (w,c_j) of Y such that $w \in \{w | w \in B_i$ and $w \notin \{c_1,\cdots,c_t\}\}$ and $c_j \in \{c_1,\cdots,c_t\}$ **do**
17: Set $Y \leftarrow$ SUB $(G_3, X\cap B_i, Y, B_i, c_j, w)$.
18: **end while**
19: **end for**
20: **end if**
21: Let k be the number of cut-vertices in P. Pick a leaf-block path $P=\{\tilde{B}_0,\cdots,\tilde{c}_i,\tilde{B}_i,\cdots\}$, where $\tilde{B}_0,\tilde{c}_i$, and $\tilde{B}_i$, are a leaf sub-block, a cut-vertex, and an internal sub-block of T respectively such that in Y there is no path from any $\tilde{B}_i$ to $Y-B$ for $i<k$. Suppose $P_i = P - \{\tilde{B}_0,\cdots,\tilde{c}_i\}$.
22: Set FIN $\leftarrow$ NOT-OK.
23: **for** $i=0$ to $k-2$ **do**
24: **if** there is an H_3-path H' from $\tilde{B}_i$ to $\tilde{B}_{i+1}$ **then** set $Y \leftarrow H \cup H'$ and FIN $\leftarrow$ OK. Finally, exit this loop.
25: **else** Find a H_3-path H' from $\tilde{B}_i$ to $(Y-P)\cup P_{i+1}$ and another H_3-path H'' from $\tilde{B}_{i+1}$ to $P-P_i$.
26: **if** such H' and H'' exist **then** set $Y \leftarrow Y \cup H' \cup H''$ and FIN $\leftarrow$ OK. Finally, exit the for-loop.
27: **end if**
28: **end if**
29: **end for**
30: **if** FIN $\neq$ ok **then** /* $\tilde{c}_k$ will become a good point. */
31: **if** P ends at $\tilde{B}_k$ **then** find an H_3-path H' from $\tilde{B}_{k-1}$ to $Y-(P-\tilde{B}_k)$ and set $Y \leftarrow Y \cup H'$.
32: **else** /* P ends at $\tilde{c}_k$ */ Note that $Y-\{\tilde{c}_k,v\}$ can be divided into at most five parts. Find at most four H_3-paths with length no more than three to make $Y-\{\tilde{c}_k,v\}$ connected. Add the H_3-paths to Y.
33: **end if**
34: **end if**
35: Return Y.
36: **end function**

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

some of $\tilde{B}_{m''}$ ($m' \neq m''$), then there exists a path a path $\tilde{Q}_{m''}$ (which is independent of $\tilde{P}_{m''}$) in $\tilde{B}_{m''}$ connecting $\tilde{c}_{m''-1}$ and $\tilde{c}_{m''}$. Replace $\tilde{Q}_{m'}$ (resp. $\tilde{Q}_{m''}$) with $\tilde{P}_{m'}$ (resp. $\tilde{Q}_{m''}$) in the original path P. Then x and y is still connected to each other in $H - \{v\}$ after the deletion of u and w. In case that one of u and w is contained in some of $\tilde{B}_m$, it is not difficult to see that the same result still holds.

Case 2: $w \in B_j$ ($j \neq i$), $w \notin B_i$ and $w \in \{c_1, \cdots, c_t\}$. Note H is 2-connected. It follows that there exists another path $P' = (xP'_1c'_1P'_2c'_2 \cdots, c'_pP'_py)$ connecting x and y, which is independent of the previous path P and passes through vertex v. If u is not contained in the path P', clearly x and y is still connected through P' after the deletion of u in $H - \{w\}$. Otherwise, suppose u is contained in P'. Similar to Case 1, we can replace one P'_m with another independent path Q'_m in B_i such that after the deletion of u, the vertices x and y still connects each other in $H - \{w\}$.

Thus x and y is always connected with each other in $H - \{u, w\}$ for $x, y \in V(H) - \{v, u, w\}$. Moreover, if one of x and y equals v, the lemma can be easily shown to be true. Therefore, the lemma holds. ■

Lemma 4.2: After the line 13 of Algorithm 1 is executed, then u becomes a good-point.

Proof: Suppose we confront the Line 13 of Algorithm 1 after level-l decomposition, where $l \geq 1$, $T_l = \{B_1\}$. This means that in block B_1, there is no point except v can make a pair of separator with node u. Note that B_1 is a 2-connected block, $B_1 - \{u\}$ remains connected as an induced subgraph. However, $\{v, u\}$ is a separator implies that there exists an H-path H' from $B - \{v, u\}$ to $Y - B$. Once H' is added, $\{u, v\}$ is no longer a separator any more. Next, we show that u and w cannot be a pair of separator for any $w \in Y - B$.

Suppose that w is contained in a block B' in the multi-level decomposition tree T_i with the maximum index i. Since B_1 lies in the bottom-most of the tree, u is also contained in some block B'' which is in T_i. If $B' \neq B''$, then according to Lemma 4.1, u and w cannot be a separator since u is also an internal node of B''. Now, suppose that u and w lie in the same block $B' = B''$ in T_i. Note u is still contained in a sub-block B''' of B' in the $(i+1)$-th level, which is obtained from B' by deleting a node. Clearly, w must be the deleted node, otherwise w will be contained in a block with level $i + 1$; a contradiction to the maximality assumption. Again, using Lemma 4.1, u and w cannot be a pair of separator. Thus, u cannot constitute a separator with any other nodes in H, i.e., u becomes good. The lemma holds. ■

Lemma 4.3: After Lines 22-34 of Algorithm 1 are executed, at least one $\tilde{c}_i \in X$ becomes a good point.

Proof: Since $s \geq 2$, there must exist leaf-block path $P = \{\tilde{B}_0, \tilde{c}_1, \tilde{B}_1, \cdots, \tilde{c}_i, \tilde{B}_i, \cdots\}$ such that no existing paths in Y connecting each $\tilde{B}_i$ to $Y - B$, since otherwise no $\tilde{c}_i$ and v constitute a separator, which contradicts to the assumption that v and some internal nodes in B form a separator. In what follows, we distinguish two cases.

Case 1: Lines 22-29 of Algorithm 1 is executed. We have a leaf-block path $P = \{\tilde{B}_0, \tilde{c}_1, \tilde{B}_1, \tilde{c}_2, \cdots, \tilde{c}_k, \tilde{B}_k\}$, or $P = \{\tilde{B}_0, \tilde{c}_1, \tilde{B}_1, \tilde{c}_2, \cdots, \tilde{c}_k\}$. In the first case, whenever there exists an H_3-path H' connecting $\tilde{B}_i$ and $\tilde{B}_{i+1}$ in Line 22-29. Then $\tilde{c}_{i+1}$ is a good-point. Actually, let B be the block including $\{\tilde{B}_1, \cdots, \tilde{B}_s\} \cup \{\tilde{c}_1, \cdots, \tilde{c}_t\}$ and v. Similar to the proof of Lemma 4.2, $\tilde{c}_{i+1}$ cannot constitute a pair of separator with any other points in $Y - B$. On the other hand, inside the block B, $\tilde{c}_{i+1}$ cannot constitute a pair of separator with any other points in $\tilde{B}_i$ and $\tilde{B}_{i+1}$ due to the algorithm, and $\tilde{c}_{i+1}$ cannot constitute a pair of separator with any other points in $B - (\tilde{B}_i \cup \tilde{B}_{i+1})$ because after the deletion of $\tilde{c}_{i+1}$, B remains 2-connected. Thus $\tilde{c}_{i+1}$ is a good point. While Line 23 is executed. Similar result holds.

Case 2: Lines 30-34 of Algorithm 1 is executed: In case that Line 31 is executed, the proof is similar to Case 1. If line 32 is executed, note v and $\tilde{c}_k$ is a pair of separator of Y, then $Y - \{v, \tilde{c}_k\}$ is divided into at most five parts. Thus, at most four H_3-paths are needed to make $Y - \tilde{c}_k$ connected. After adding these H_3-paths, B remains 2-connected. Similar arguments as Case1 can be used to show $\tilde{c}_k$ has become good. ■

Lemma 4.4: At least one case is true in Lines 22-34 of Algorithm 1.

Proof: First suppose that the leaf-block path $P = \{\tilde{B}_0, \tilde{c}_1 \cdots, \tilde{c}_i, \tilde{B}_i, \cdots, \tilde{c}_k, \tilde{B}_k\}$, where $\tilde{B}_k$ is the block with degree at least three in the leaf-block tree. Next, we use case by case analysis to show that at least one case is true in lines 22-34 of Algorithm 1.

Consider $\tilde{B}_1$. Note that $\{\tilde{c}_1, \tilde{c}_2\}$ is a pair of separator of Y. It follows that there exists an H_3-path H'_1 connecting $\tilde{B}_1 - \{\tilde{c}_1, \tilde{c}_2\}$ and $Y - \tilde{B}_1$. If the H_3-path H'_1 connects $\tilde{B}_0 - \{\tilde{c}_1\}$ and $\tilde{B}_1 - \{\tilde{c}_1\}$, then it is done. Otherwise, there exists an H_3-path H'_1 starting at $\tilde{B}_1 - \{\tilde{c}_1, \tilde{c}_2\}$ ending at $(Y - P) \cup P_2$.

Now consider $\tilde{B}_2$. Note that $\{\tilde{c}_2, \tilde{c}_3\}$ is a pair of separator of Y. It follows that there exists an H_3-path H'_2 connecting $\tilde{B}_2 - \{\tilde{c}_2, \tilde{c}_3\}$ and $Y - \tilde{B}_2$. If H'_2 connects $\tilde{B}_2 - \{\tilde{c}_2, \tilde{c}_3\}$ and $P - P_1$ or $\tilde{B}_1 - \{\tilde{c}_2\}$, then it is done. Otherwise, there exists an H_3-path H'_2 starting at $\tilde{B}_2 - \{\tilde{c}_2, \tilde{c}_3\}$ ending at $(Y - P) \cup P_3$.

Generally, since $\{\tilde{c}_i, \tilde{c}_{i+1}\}$ is a pair of separator of Y. It follows that there exists an H_3-path H'_i connecting $\tilde{B}_i - \{\tilde{c}_i, \tilde{c}_{i+1}\}$ and $Y - \tilde{B}_i$. If H'_i connects $\tilde{B}_i - \{\tilde{c}_i, \tilde{c}_{i+1}\}$ and $P - P_{i-1}$ or $\tilde{B}_{i-1} - \{\tilde{c}_i\}$, then it is done. Otherwise, there exists an H_3-path H'_i starting at $\tilde{B}_i - \{\tilde{c}_i, \tilde{c}_{i+1}\}$ ending at $(Y - P) \cup P_{i+1}$.

Continuing this process$\cdots$ Finally, for $\tilde{B}_{k-1}$, there exists an H_3-path H'_{k-1} connecting $\tilde{B}_{k-1} - \{\tilde{c}_{k-1}, \tilde{c}_k\}$ and $Y - \tilde{B}_{k-1}$. If H'_{k-1} connects $\tilde{B}_{k-1} - \{\tilde{c}_{k-1}, \tilde{c}_k\}$ and $P - P_{k-2}$ or $\tilde{B}_{k-2} - \{\tilde{c}_{k-1}\}$, then it is done. Otherwise, there exists an H_3-path H'_{k-1} starting at $\tilde{B}_{k-1} - \{\tilde{c}_{k-1}, \tilde{c}_k\}$ ending at $(Y - P) \cup P_k = (Y - P) \cup B_k$.

Since the above process have enumerated all possible cases, therefore at least one case is true. For the case that the leaf-block path $P = \{\tilde{B}_0, \tilde{c}_1 \cdots, \tilde{c}_i, \tilde{B}_i, \cdots, \tilde{c}_k\}$, a similar argument can still be applied. This completes the proof. ■

Lemma 4.5: Each time, Line 4 of Algorithm 1 is executed, at least one bad point in X will become good points by adding at most 12 nodes, where X is the number of bad points in $Y := C_{2,3}$.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Proof: By Lemmas 4.2-4.4, at each iteration, at leat one bas point becomes good by adding at most 4 H_3-paths, the length of which is at most three. Therefore, at most 12 nodes are needed to changed one bad-point into a good-point. ■

Theorem 4.6 ([10]): There exists a polynomial time approximation algorithm A which can generate an approximated solution for $(2, m)$-CDS with performance ratios $(5 + \frac{25}{m})$ for $2 \leq m \leq 5$ and 11 for $m > 5$.

Lemma 4.7: Given a G_3, let H be a connected subgraph containing a $C_{2,3}$ of the G_3. Then, adding a H_3-path P to H does not introduce a new bad-point to H.

Proof: To prove this, we consider following two cases in which the length of P is two or three hops. In the first case, P includes only one new node u. Then, no node in H can constitute a separator with u in $H \cup \{u\}$ since H is 2-connected. Therefore, all nodes in P are good-points in $H \cup \{u\}$. Now, we consider the second case in which P has two new points u and v. Now, we claim that u is not a bad-point in $H \cup \{u, v\}$. Suppose u is a bad-point and it constitute a separator with another node $w \in H \cup \{v\}$. Clearly, $w \neq v$ since H is 2-connected. On the other hand, since H is 2-connected and v has at least three neighbors in H, $H \cup \{v\}$ is at least 2-connected. Therefore, $(H \cup P) - \{u, w\}$ is connected for any $w \in H$. Therefore, u is not a bad-point and by the same argument, v is not a bad-point. In conclusion, the lemma is true. ■

Theorem 4.8: Algorithm 1 is a $\frac{520}{3}$-approximation for 3-Connected 3-Dominating Set problem.

Proof: From Theorem 4.6, we have a r-approximation algorithm for computing a $C_{2,3}$, where $r = \frac{40}{3}$. Then, we can have a $C_{2,3}$ such that $|C_{2,3}| \leq r|C^{opt}_{2,3}|$. In Algorithm 1, since $X \subseteq C_{2,3}$, $|X| \leq |C_{2,3}|$. From Lemma 4.5 and Lemma 4.7, Algorithm 1 will use at most $12|X|$ nodes to augment the $C_{2,3}$ to a $Y = C_{3,3}$. As a result, the size of final Y is bounded by $|Y| = |C_{2,3}| + 12|X| \leq r|C^{opt}_{2,3}| + 12|C_{2,3}| \leq r|C^{opt}_{2,3}| + 12r|C^{opt}_{2,3}| \leq 13r|C^{opt}_{2,3}| \leq 13r|C^{opt}_{3,3}|$. ■

C. Generalization for any $m \neq 3$

When $m > 3$, we first compute a $C_{2,m}$ using the existing algorithm in [10], [11]. Then, we augment $C_{2,m}$ to $C_{3,m}$ using Algorithm 1. Now, we prove that the size of the outputs by this strategy is within a constant factor from an optimal solution even in the worst case.

Theorem 4.9: The PR of this strategy is $13r$ for $m > 3$, where $r = (5 + \frac{25}{m})$ for $3 \leq m \leq 5$ and 11 for $m > 5$

Proof: For $m > 3$, it is easy to show that $|Y| \leq 13r|C^{opt}_{3,m}|$ using the argument in the proof of Theorem 4.8. ■

When $m = 1, 2$, We start from a $C_{1,3}$, then augment $C_{1,3}$ to $C_{2,3}$. Both can be computed by the existing method in [10], [11]. Finally, augment $C_{2,3}$ to $C_{3,3}$ using Algorithm 1. Now, we evaluate the worst case quality of outputs of this approach.

Theorem 4.10: The PR of this strategy is $17 \cdot 2 \cdot 12$ for $m = 1, 2$.

Proof: First, we focus on the case $m = 1$, and show that the obtained $C_{3,3}$ is within a constant factor from $C^{opt}_{3,1}$. In fact, by the Algorithms in [10], [11], we have $C_{1,3} = I_1 \cup I_2 \cup I_3 \cup C$, where I_i is the maximal independent set obtained sequentially and C is some additional nodes added to make I_1 to be connected. Since $|I_i| \leq 5|C^{opt}_{1,1}|$ $(i = 1, 2, 3)$ and $|C| \leq 2|I_1|$, we get we have $|C_{1,3}| \leq 17|C^{opt}_{1,1}|$. Moreover, by the proofs in [10], [11], we have $|C_{2,3}| \leq 2|C_{1,3}|$. By Algorithm 1, $|C_{3,3}| \leq 12|C_{2,3}| \leq 2 \cdot 12|C_{1,3}| \leq 17 \cdot 2 \cdot 12|C^{opt}_{1,1}|$. Note that $|C^{opt}_{1,1}| \leq |C^{opt}_{3,1}|$. It follows that $|C_{3,3}| \leq 17 \cdot 2 \cdot 12|C^{opt}_{3,1}|$.

For $m = 2$, the same algorithm as above can be applied. And the approximation ratio can be obtained similarly by noting that $|C^{opt}_{1,1}| \leq |C^{opt}_{3,2}|$. ■

By combining Theorem 4.8, Theorem 4.9, and Theorem 4.10, we have the following conclusion.

Theorem 4.11: There exists an $O(1)$-approximation algorithm for computing $(3, m)$-CDS in UDGs for any m.

V. Conclusion

In this paper, we studied the problem of constructing fault-tolerant CDSs in homogeneous wireless networks, which can be abstracted as the minimum k-connected m-dominating set problems. We observe every existing approximation algorithm for this problem is flawed for $k \geq 3$ in a sense that it does not generate a feasible solution in some UDGs. Based on the observations, we propose a constant factor polynomial time approximation algorithm to compute $(3, m)$-CDSs. As a future work, we are interested in generalizing our algorithm for any $k \geq 4$.

VI. Acknowledgement

This work was supported in part by the US National Science Foundation (NSF) Grant No. CCF-0514796 and CCF-0627233. The research is also supported in part by NSFC (10971255), the Key Project of Chinese Ministry of Education (208161), and Program for New Century Excellent Talents in University. The authors would like to appreciate to Prof. Feng Wang from Arizona State University, Prof. My T. Thai from University of Florida, and Prof. Yingshu Li and Yiwei Wu from Georgia State University for their valuable comments on the counter examples.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, pp. 102-114, August 2002.

[2] C. R. Dow, P. J. Lin, S. C. Chen, J. H. Lin, and S. F. Hwang, "A Study of Recent Research Trends and Experimental Guidelines in Mobile Ad Hoc Networks," *the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*, pp. 72-77, Taiwan, March 2005.

[3] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99)*, pp. 152-162, Washington, USA, August 1999.

[4] P. Sinha, R. Sivakumar, and V. Bharghavan, "Enhancing Ad Hoc Routing with Dynamic Virtual Infrastructures," *the 20th annual joint conference of the IEEE computer and communications societies*, vol. 3, pp. 1763-1772, 2001.

[5] A. Ephremides, J. Wieselthier, and D. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," *Proceeding of the IEEE*, vol. 75, issue 1, pp. 56-73, 1987.

[6] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica*, vol. 20, pp. 374-387, April 1998.

[7] M.R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-completeness," Freeman, San Francisco, 1978.
[8] D. Kim, X. Li, F. Zou, Z. Zhang, and W. Wu, "Recyclable Connected Dominating Set for Large Scale Dynamic Wireless Networks," *The 3rd International Conference on Wireless Algorithms, Systems and Applications (WASA 2008)*, Dallas, TX, USA, Oct. 26-28, 2008.
[9] F. Wang, M. T. Thai, D.-Z. Du, "2-Connected Virtual Backbone in Wireless Network," *IEEE Transactions on Wireless Communications*, vol.8, no. 3, pp. 1230-1237, Mar. 2009.
[10] W. Shang, F. Yao, P. Wan, and X. Hu, "On Minimum m-Connected k-Dominating Set Problem in Unit Disc Graphs," *Journal of Combinatorial Optimization*, Dec. 2007.
[11] M.T. Thai, N. Zhang, R. Tiwari, and X. Xu, "On Approximation Algorithms of k-Connected m-Dominating Sets in Disk Graphs," *Theoretical Computer Science*, vol. 358, pp. 49-59, 2007.
[12] M.L. Huson and A. Sen, "Broadcast Scheduling Algorithms for Radio Networks," *IEEE Military Communications Conference (MILCOM '95)*, vol. 2, pp. 647-651, 1995.
[13] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs", *Discrete Mathematics*, vol. 86, pp. 165-177, Dec. 1990.
[14] R. Diestel, "Graph Theory 3rd ed.," *Graduate Texts in Mathematics*, vol. 173, Springer-Verlag, Heidelberg, 2005.
[15] F. Dai and J. Wu, "On Constructing k-Connected k-Dominating Set in Wireless Network," *the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.
[16] N. Zhang, I. Shin, F. Zou, W. Wu, and M.T. Thai, "Trade-off Scheme for Fault Tolerant Connected Dominating Sets on Size and Diameter," *ACM International Workshop on Foudations of Wireless Ad Hoc and Sensor Networking and Computing (FOWANC)*, in conjunction with MobiHoc, 2008.
[17] Y. Wu, F. Wang, M.T. Thai, and Y. Li, "Constructing k-Connected m-Dominating Sets in Wireless Sensor Networks", *2007 Military Communications Conference (MILCOM07)*, Orlando, FL, October 29-31, 2007.
[18] Y. Wu and Y. Li, "Construction Algorithms for k-Connected m-Dominating Sets in Wireless Sensor Networks," *9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc 2008)*, Hong Kong, China, May 26-30, 2008.
[19] W. Wu, H. Du, X. Jia, Y. Li, S.C.-H. Huang, "Minimum Connected Dominating Sets and Maximal Independent Sets in Unit Disk Graphs," *Theoretical Computer Science*, vol. 352, 2006.
[20] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, New York, USA, June 2002.
[21] D.S. Johnson, "Approximation Algorithms for Combinatorial Problems," *Journal of Computer System Science*, vol. 9, pp. 256-278, 1974.
[22] K. Menger, "Zur allgemeinen Kurventheorie," *Fundamenta Mathematicae*, vol. 10, pp. 96-115, 1927.
[23] M. T. Thai, F. Wang, D. Liu, S. Zhu and D-Z. Du, "Connected Dominating Sets in Wireless Networks with Different Transmission Ranges," *IEEE Transactions on Mobile Computing*, Vol. 6, No. 7, July 2007.
[24] Y. Li, S. Zhu, My T. Thai, and D-Zhu Du, "Localized Construction of Connected Dominating Set in Wireless Networks," *NSF International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks (TAWN04)*, Chicago, June 2004.

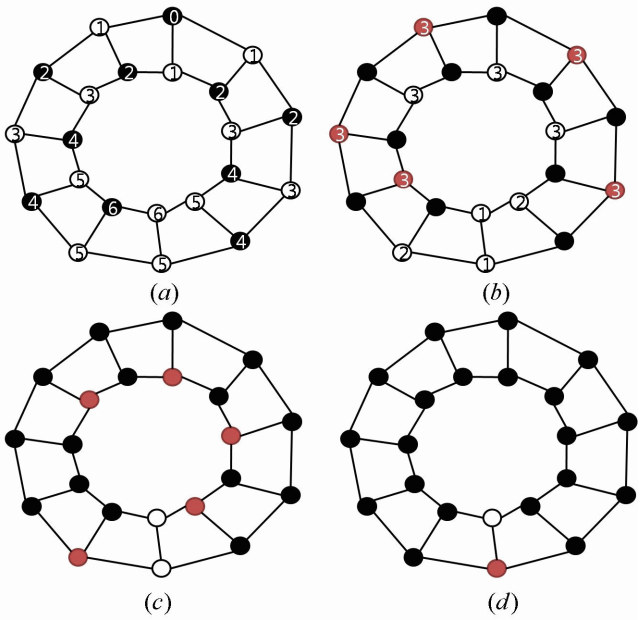

Fig. 3. Figure (a) to (d) illustrate a situation in which CDSAN does not work correctly. Note that the figure is little bit crude, but clearly its magnified version (with more nodes) can be embedded in UDG. In the figure (a), a 3-connected UDG G is given and our goal is computing a $(3,3)$-CDS of G. One apparent solution can be obtained by selecting whole nodes. Based on the description of the algorithm in [11], we pick one root node and compute the hop distance from the root to each node. Then, we follow the MIS construction method in [20]. In the figure, the number in each node is their hop distance from the root and the set of black nodes are MIS nodes. In the figure (b), the TFA algorithm in [23] is applied and the MIS nodes are connected such that it selects non-MIS nodes with highest number of MIS neighbors and colors it grey until the MIS nodes are fully connected. In the figure, the number in each node represents the number of adjacent MIS nodes. In the figure (c) and (d), CDSMIS is applied and based on the $(1,1)$-CDS computed by the TFA algorithm, we have $(1,3)$-CDS. Note that the resulting subgraph induced by colored nodes is 2-connected and contains no 3-connected subgraph. Now, CDSAN needs to identify at least one 3-connected subgraph but there is no such subgraph. Therefore, C, which is returned by CDSAN, is a $(2,3)$-CDS.

APPENDIX

In Appendix, we investigate every existing approximation algorithm for computing (k,m)-CDSs, where $k \geq 3$ and discuss about their limitations.

A. CDSAN: Connected Dominating Set by Adding Nodes for k-CDSs [11]

In [11], the authors first present the CDSAN for computing (k,k)-CDSs and based on this result, built an approximation algorithm to generate (k,m)-CDSs. The CDSAN uses a $(1,m)$-CDS as its input and increases its connectivity using an idea similar to Wang et al.'s in [9]. That is, if we have a k-connected subgraph, it identifies every $(k+1)$-connected block in the subgraph first. As long as there are more than one $(k+1)$ connected block, it sets a path from a $(k+1)$-connected block to the rest part of k-connected subgraph so that the $(k+1)$-connected block can be augmented. Unfortunately, such generalization does not work for any k. If $k=1$, as shown in [9], we always have at least one 2-connected block if the size of a given graph is more than two. However, this is not true when $k \geq 2$ (See Figure 3).

Since CDSAN does not work for some UDGs, their (k,m)-CDS construction algorithm based on CDSAN does not work for some UDGs. Furthermore, since UDG is a subset of DGB, CDSAN is not working perfectly for DGBs. In [16], Zhang et al.'s built an approximation algorithm to compute (k,m)-CDSs with bounded diameter in DGBs. Based on our result, one can easily see that this algorithm also does not work correctly for some UDGs.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

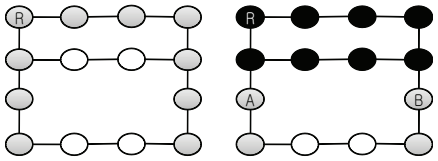

Fig. 4. This figure illustrates a situation in which DDA does not work correctly. Suppose we want to compute a $(2,1)$-CDS. Observe that the given graph is 2-connected. In the left figure, a set of grey nodes forms a $(1,1)$-CDS. In the right figure, R is a random root and it searches a 2-connected subgraph, which is the set of black nodes, C. Now, to expand the 2-connected subgraph, we can start from node A or B. However, we cannot find a path P such that all the nodes in P are adjacent to at least one black node and either $C \cup P \cup \{A\}$ or $C \cup P \cup \{B\}$ is a 2-connected subgraph. Therefore, we cannot use Lemma 6.1 to augment the 2-connected subgraph and the algorithm fails.

B. DDA: Distributed Deterministic Algorithm for (k,m)-CDSs [17]

Now, we discuss about DDA which is a distributed approximation algorithm to compute (k,m)-CDSs in UDGs [17]. Briefly, DDA consists of following three phases: In the first phase, it uses an existing distributed 1-CDS algorithm in [24] to generate a $C_{1,1}$. In the second phase, the $C_{1,1}$ evolves to a $C_{1,m}$ by using the idea in [10]. In the third phase, DDA makes the $C_{1,m}$ to be k-connected by adding some nodes using the idea in Lemma 6.1.

Lemma 6.1 ([17]): Let G be a k-connected graph and v be a node not in G having at least k-neighbors in G. Then, $G' = G \cup \{v\}$ is also a k-connected graph.

In detail, the DDA works as follows: A random root r collects information of nodes within i hops where i increases from 2 until r can find a k-connected subgraph of G, which is a given UDG, using the information. Once a k-connected subgraph of G, say C, is found by r, DDA finds a path from a node $x \in C_{1,m}$ to $C - \{x\}$ such that x and all nodes in the path are adjacent to at least one node in C. Once a node in the path receives multiple request to be on a path for more than k nodes, it declares that it will join to the k-connected subgraph. x and a path for it can join to C once all the node in the path declare to join. In this way, C is expanded. This algorithm can successfully generates a (k,m)-CDS for some graph. However, there are some cases where DDA fails. That is, there can be a k-connected graph in which Lemma 6.1 cannot be applied (See Figure 4).

C. LDA: Distributed Local Decision Algorithm for (k,m)-CDSs [18]

In [18], the authors introduce the LDA, a localized algorithm to compute (k,m)-CDSs for UDGs. LDA consists of three phases. In the first phase, it computes a $C_{1,m}$ using similar idea for DDA in [17]. Note that the $C_{1,m}$ includes a $C_{1,1}$. In the second phase, each pair of adjacent nodes in the $C_{1,1}$ negotiates with each other. That is, two adjacent nodes x and y in the $C_{1,1}$ try to color at least $k-2$ common neighbors in black. Now, define a local graph of a vertex v as a graph induced by v and its neighbors. Then, in the third step, each black node in the $C_{1,1}$ builds a local k-connected graph, which includes all the black neighbors of v in the $C_{1,m}$ and negotiated neighbors of v in the second step and color whole nodes in the k-connected local graph in black. Authors claim that at the end of this phase, the set of black nodes is a (k,m)-CDS. However, in some G_k, some black node in $C_{1,1}$ may not have a local k-connected graph and LDA does not produce a (k,m)-CDS correctly (See Figure 5).

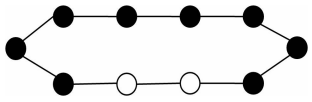

Fig. 5. This figure illustrates a situation in which LDA does not work correctly. Suppose our goal is computing a $(2,1)$-CDS. Then, LDA first computes a (1,1)-CDS, which is the set of black nodes. In the second phase, LDA makes no change on the set. In the third phase, no node in the $C_{1,1}$ can construct a 2-connected local graph and the result of LDA over this graph is a 1-connected subgraph. However, the given graph is 2-connected and includes a feasible solution.

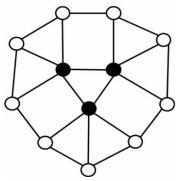

Fig. 6. This figure illustrates a situation in which ICGA does not work correctly. In this figure, the whole graph is 3-connected and G, which is the set of black nodes, is 2-connected. Then, we have no F which is a connected graph and 2-dominates G. Therefore, ICGA does not work in this case.

D. ICGA: Centralized Algorithm for (k,m)-CDSs [18]

In [18], the authors introduce ICGA, a centralized algorithm to compute (k,m)-CDSs for UDGs. This algorithm introduces a new approach which is very different from previous ones. The core strategy of ICGA is that it first computes a $(1,m)$-CDS and tries to evolve it to a (k,m)-CDS by using the idea in Lemma 6.2.

Lemma 6.2: Given a k-connected graph G and a connected set F which can k dominate G, the graph G' composed by $G \cup F$ is $(k+1)$-connected.

ICGA starts from a 2-connected subgraph in $(1,m)$-CDS and finds F which k dominates the subgraph, which means that each node in F has at least k neighbors in G. By repeatedly applying Lemma 6.2, the authors claim that ICGA generates a (k,m)-CDS finally. To prove the correctness of Lemma 6.2, the authors assume that when they have a k-connected subgraph G and a connected set F which k dominates G, deleting $k-1$ nodes from G or F cannot disconnect $G \cup F$. However, they missed that there is a 3-connected graph in which this lemma is not true (See Figure 6).