Randomized Algorithms for Approximating a Connected Dominating Set in Wireless Sensor Networks

Akshaye Dhawan, Michelle Tanco, Aaron Yeiser

Department of Mathematics and Computer Science
Ursinus College
Collegeville, PA
adhawan@ursinus.edu, mitanco@ursinus.edu, aayeiser@ursinus.edu

Abstract

A Connected Dominating Set (CDS) of a graph representing a Wireless Sensor Network can be used as a virtual backbone for routing through the network. Since the sensors in the network are constrained by limited battery life, we desire a minimal CDS for the network, a known NP-hard problem. In this paper we present three randomized algorithms for constructing a CDS. We evaluate our algorithms using simulations and compare them to the two-hop K2 algorithm and two other greedy algorithms from the literature. After pruning, the randomized algorithms construct a CDS that are generally equivalent in size to those constructed by K2 while being asymptotically better in time and message complexity. This shows the potential of significant energy savings in using a randomized approach as a result of the reduced complexity.

Keywords—Distributed Computing; Wireless Sensor Networks; Dominating Sets;

I. Introduction

Constructing a dominating set is a key approach to creating a backbone for data gathering and communication in a Wireless Sensor Network [1] [2]. In this paper, we use a graph $G = (V, E)$ to represent the wireless sensor network, where V is the set of sensors in the network and an edge $(u, v) \in E$ represents a link between two sensors u, v that are within communicating distance of each other. We also assume that all sensors are deployed on a 2-dimensional plane and have a uniform transmission range.

Given such a representation of a sensor network, a *dominating set* (DS) of a graph G is a subset $D \subset V$ such that for all $u \in V$ either $u \in D$ or u is adjacent to a node in D (i.e., $(u, w) \in E$ for some $w \in D$). Nodes in the dominating set D are referred to as *dominators* and the remaining nodes in $V - D$ are referred to as *dominatees*. A variation to this problem is that of the Connected Dominating Set (CDS) which can be defined as a set that is dominating and induces a connected subgraph. In other words, it is a set of nodes $C \subset V$ such that the nodes in C are both dominating and connected.

The construction of a CDS provides the network with a virtual backbone over which routing, multicast and broadcast can be performed since every node is either in the backbone or has a neighbor in the backbone. Also, the construction of a CDS, allows the network to adapt to changes in the topology since only the nodes in the CDS need to be aware of routing information. By being connected the backbone can relay a message to either the destination directly (if the destination is in the CDS) or through the dominator of the destination. Since the nodes in the CDS are actively draining their batteries by serving as relay nodes for the network, it is desirable to construct a minimum size CDS. However this has been shown to be a NP-hard problem in [3]. Numerous centralized and distributed algorithms have been presented in the literature with the size of the CDS they construct being a key performance metric. This is because the greater the size of the CDS, the more nodes there are participating in the network. As a result of this, the nodes in the CDS burn more energy. Hence, a smaller size CDS is desirable.

In this paper, we present and assess three new randomized algorithms. We compare them to the 2-phase distributed algorithm K2 [4] and to another 2-phase greedy algorithm called GreedyConnect [5]. Although the randomized algorithms initially create large dominating sets, their performance can be improved significantly by utilizing the second phase of K2 to trim unnecessary sensors from the set. This results in the randomized algorithms having very similar performance to K2 with a much lower time and message complexity since they are more efficient

in Phase 1. They are slightly worse in performance to GreedyConnect but the larger CDS size may be a worthy tradeoff for energy savings resulting from a lower time and message complexity for the randomized algorithm. Our simulations also showed that for networks of a given density, the randomized algorithm could be repeated at each sensor a certain number of times and would result in a CDS with a very high probability. This allowed us to implement a distributed version of the algorithm. The most efficient of our algorithms had a time and message complexity of $O(n)$, where n is the size of the network when compared to a time complexity of $O(\Delta^2)$ and a message complexity of $O(\Delta)$ for Phase 1 of K2, where Δ is the maximum degree in the network.

The remainder of this paper is organized as follows. In Section II we present related work on this problem and introduce the K2 and GreedyConnect algorithm in some detail since we use ideas from both algorithms in our algorithms. Section III presents a centralized randomized algorithm and two distributed greedy algorithms for constructing a CDS. Our simulation studies comparing the three randomized algorithms to K2 and GreedyConnect are presented in Section IV. Finally, we conclude in Section V.

II. Related Work

In this section we briefly summarize the literature on connected dominating sets for sensor networks. Both distributed and centralized algorithms are presented in the literature. As a comprehensive survey on this area, readers are referred to [6].

The applications of a connected dominating set to routing in ad hoc networks were first outlined in [7] where they presented the idea of constructing a connected dominating set and using it as a backbone in the network for routing.

Several approximation algorithms were presented in the literature. [8] presents a centralized algorithm with a $O(H\Delta)$ approximation factor where Δ is the maximum degree and H is the harmonic function. In [9] the authors present a 1-phase greedy algorithm that has a performance ration of $2 + ln\Delta$.

[10] was one of the first distributed algorithms to show an improved analysis of the relationship between the size of a maximal independent set and a minimum CDS in a unit disk graph, which yields better bounds for many other algorithms. [11] presents a distributed algorithm for CDS construction by constructing a spanning tree first and then marking every node in the tree as a dominator node or dominated node. The Performance Ratio for this algorithm was shown to be 8. In [12] the same author noticed the difficulty of maintaining a CDS and designed a localized 2-phase algorithm that uses a Maximal Independent Set but this algorithm has a PR of 192. [13] presents the best known PR of $4.8 + ln5$ in a centralized algorithm. The algorithm is known as S-MIS and uses a Steiner Tree to construct a CDS. In this algorithm they build a Maximal Independent Set in Phase 1. Then in Phase 2, they employ a greedy algorithm to construct a Steiner tree with minimal number of Steiner nodes to connect the nodes in the MIS. They mention that a distributed implementation is possible but do not elaborate on this algorithm or its PR.

[14] considers the problem of constructing a strongly connected dominating set in asymmetric wireless networks. They build on the work of [15] and present an improved algorithm for CDS construction in a digraph which has polynomial time and logarithmic approximation ratio. [16] presents an approximation algorithm based on dynamic programming for finding a minimum weighted dominating set in a node-weighted graph.

To the best of our knowledge, [17] and [18] are the only work in the literature that look at randomization. However they only use it to break ties in a greedy algorithm and to generate a random delay respectively.

A. K2 Marking and Pruning

The K2 [4] algorithm has two phases, first the construction of a connected dominating set, then the reduction of unnecessary vertices from the CDS. In phase 1, each sensor adds itself to the dominating set if any two of its neighbors are not neighbors. It is clear that if we start with a connected graph we will get a CDS since any two non-connected sensors that share a neighbor will be connected. This phase has a time complexity of $O(\Delta^2)$ where Δ is the maximum degree in the graph. In a dense graph with n nodes, $\Delta = O(n)$.

However, this set is likely to contain many more nodes than necessary since the marking process was very simple. In order to reduce the size of the set, the authors use a k-reduction (where k is the number of hops the algorithm is looking at) to remove unnecessary sensors from the set. For each sensor in the dominating set we consider every k-hop group of neighbors where each member of the group is in the dominating set. If one of these groups contains every neighbor of the original sensor in its neighbor set, we remove the original sensor from the dominating set. The dominating set is still connected by the group of k neighbors. We call this the K2 algorithm because we compare ourselves against the 2-hop version of this algorithm.

As k increases the size of the connected dominating set decreases. However, the message and time complexity does increase because we have to check each size k group of neighbors for each sensor. For the purposes of this paper we let $k = 2$ as larger values of k have a message and time complexity which would drain the network more it would benefit it by creating a path for routing. This algorithm has

the benefit of the CDS being easy to maintain. Combining both phases, we get a time complexity of O(Δ^2) and a message complexity of O(Δ).

B. Greedy Connect

In [5] the authors introduced a 2-phase greedy algorithm for constructing a CDS and showed that it constructed a CDS smaller in size to K2. We now briefly describe this work.

Phase 1: At the start of the algorithm, each sensor is initialized to the color white (uncovered). Each sensor broadcasts its ID thereby allowing sensors to discover their neighbor set. Sensors then broadcast their white neighbor count. If a sensor discovers that its white count is highest or tied highest, it adds itself to the CDS. This heuristic is greedy because it selects sensors who will maximize the number of white (uncovered) sensors covered at a specific stage of the algorithm. Once added to the dominating set, a sensor colors itself black (dominating) and colors all of its neighbors grey (covered but not dominating). This denotes that those sensors are adjacent to a node in the dominating set. When a sensor's color is updated, it broadcasts this to its neighborhood so that they can update their white neighbor counts. A second pass ensures that every sensor has been dominated.The result of phase 1 is that each sensor is either in or adjacent to the dominating set.

Phase 2: Phase 2 connects the dominating components formed by phase 1. The algorithm starts by initializing the component id of each dominator to their unique id number and each dominatee to -1. Dominators then recursively share the highest component number in their neighborhood such that each dominating component will have a unique identifier - the highest sensor id of that component. Next, if a grey node discovers that two of its neighbor have different component numbers, it will add itself to the CDS and update the now connected two components to share one identifier. From a lemma in the paper, we know that each component is at most two hops way. Dominatees then share the component they are adjacent to with their neighbors. Two sensors add themselves to the CDS if they discover they are each adjacent to different components.

From phase one and two we see that the time complexity of the GreedyConnect algorithm is $O(n^2)$. Also, the message complexity is $O(n^2)$.

III. Randomized Algorithms

In this section we present three randomized algorithms: one for dominating set construction and two for connected dominating set construction.

A. Notation

We assume every sensor node to have a unique identifier. The properties of a sensor node are shown in Table I. In each algorithm we also use a color scheme to track the role of a particular sensor in the CDS. We summarize the meaning of these colors in Table II.

TABLE I. Fields for a given sensor node v

Field	Meaning
v.COLOR	The current color of the sensor
v.ID	Unique identifier for the sensor
v.WhiteCount	Number of white nodes in $N(v)$

TABLE II. Node color assignments

Color	Meaning
White	Undiscovered by the Dominating Set
Grey	Dominated but has white neighbors
Black	Dominated and has no white neighbors

B. Centralized Random Selection

Our first algorithm takes a simple approach to creating a CDS. This algorithm randomly selects sensors and adds them to the future CDS. The algorithm stops selecting sensors when it sees that a CDS has formed. This is the primary reason that the algorithm is centralized - an outside source is needed to manage the construction of the CDS and determine that a CDS has been formed. The base station for the network would typically serve in this role.

This algorithm requires the CDS constructor to be aware of two sets: C which represents the CDS under construction and R with represents vertices to be potentially selected. These sets are initially empty. The manager selects a random vertex to be added to C. Each time a vertex is added to C, all of its neighbors not already dominated by nodes in C are added to R. The manager continues to select sensors from R for C until a CDS is formed. It is clear that the formed CDS is connected since only vertices adjacent to a vertex in C can be selected.

The time complexity is bounded by $O(n)$ since for each vertex $v \in V$, can be added to R only once. The message complexity for this is minimal, since the central manager is doing all calculations, the only messages passed is at the end of the algorithm to inform sensors if they are in the CDS. This is bounded by $O(n)$.

Require: $\forall\ v \in V$, v.COLOR $\leftarrow$ WHITE
$C = \phi$
$R = \phi$
Pick random $v \in V$
$R = R \bigcup v$
while C is not a CDS **do**
Pick a random u in R
$R = R - u$

$C = C \bigcup u$
u.COLOR $\leftarrow$ BLACK
for every neighbor $w \in N(u)$ **do**
if w.COLOR $==$ WHITE **then**
$R = R \bigcup w$
w.COLOR=GREY
end if
end for
end while

C. Deriving the probability for each sensor to join the DS

We derive the probability p for a sensor to be a part of the Dominating Set in the following manner. Assume that G is a d-regular graph (i.e., each vertex $v \in V$ has degree d).

Let R be a random subset of V in which each vertex appears with independent probability p. Let D be the union of R and the set of all vertices not adjacent to a vertex in R. Note that D is a dominating set.

Let X_v be the indicator for the event $v \in D$ for some sensor v. Then, $X_v = 1$ if $v \in R$ or if v and all d of its neighbors are not in R. The probability that $v \in R$ is p and the probability that v and all d of its neighbors are not in R is $(1-p)^{d+1}$. Adding these probabilities gives $E[X_v] = p + (1-p)^{d+1}$ as the expected value of $x \in D$.

We multiply this by n, the total number of vertices since this is independent, to get $E|D| = n(p + (1-p)^{d+1})$. To optimize $E|D|$ we wish to find the minimum for $E[X_v]$ by differentiating with regards to p and setting this equal to 0. We then solve for p to get $p = 1 - (d+1)^{\frac{-1}{d}}$.

While we do not have d-regular graphs in practice, this allows us to derive a theoretical value for p. Since the process described above adds all nodes not in R to D to give a dominating set, our assumption of a d-regular graph does not impact correctness. For the purpose of our algorithms, each sensor works with its local degree instead of d.

D. Probability with Rounds

Like most randomized algorithms, the premise behind our first randomized algorithm was simple: randomly add sensors until a CDS has been formed. However, since it requires an algorithm manager to check if C is a CDS, it would imply centralized control. Keeping this in mind, we experimented with designing a distributed, localized algorithm in order to have sensors randomly add themselves to the CDS.

We next attempt to use the probability derived in the previous section to design an algorithm that can converge on a CDS in a fixed number of rounds.Through our simulations we have experimentally found that there is a small value k based on the density of the network such that k iterations of randomly selecting nodes with the derived probability were sufficient every time to create a CDS. For example, when simulating several networks of one hundred sensors disbursed in a 100 by 100m region, we found that $k = 6$ flips per sensor would suffice. Although a CDS was often created after 4 rounds, in our experiments we found that 6 rounds always sufficed for this topology density - in other words every sensor flipping a coin with a probability p six times is enough to form a CDS for this network size. In practice, this algorithm would very quickly form a large CDS. The distributed nature of this algorithm is seen through each sensor tossing a coin k times simultaneously. Although the simulations are explained in detail in section IV, figure 1, shows another way to look at this experimental observation. The figure shows the variation in the number of rounds needed based on the density (as measured by the number of neighbors) of the graph. This figure was generated through experiments conducted on a few hundred randomly generated graphs. As can be seen from the figure, at higher density, a smaller number of rounds suffices. This demonstrates a very useful property of the randomized algorithm which allows us to let each sensor flip a coin p times and leaves us with a CDS with a high probability. The algorithm is shown below.

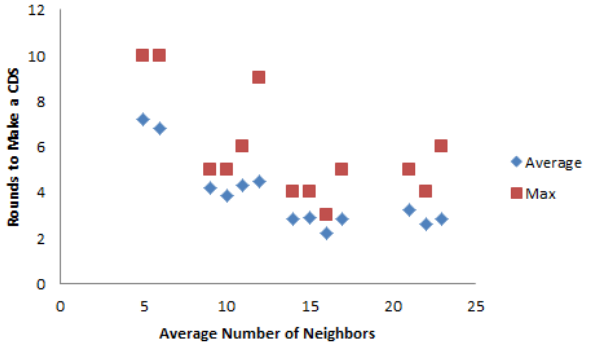

Fig. 1. Number of rounds based on density of graph

To analyze the time complexity of this algorithm we note that there are two phases. In the first phase, each sensor asynchronously broadcasts its id so that its neighbors know that it exists. During this time, each sensor can count how many neighbors it has. In the second phase, each sensor calculates the probability based on its neighbor count and flips a coin with that probability at most k times. If a sensor adds itself to the CDS, it stops tossing the coin. Since each sensor is tossing the coin independently of all other sensors the time complexity of this phase is constant.

We see that the time complexity for this algorithm is $O(k)$ where k is the number of times each sensor must repeat the random selection process. Each sensor

provides its ID to its neighborhood and then tosses a coin a constant number of times. All sensors in the network perform the decision at the same time. For analyzing the message complexity, we note that in phase one each sensor broadcasts its ID so a total of n messages are passed. At the end of the algorithm sensors that have added themselves to the CDS would broadcast this information to inform their neighbors. Hence there is a message complexity of $O(n)$. As a tradeoff for minimal time and message complexity, this algorithm produces a larger CDS. On average, it is at least twice the size of the GreedyConnect algorithm.

Require: $\forall\ v \in V$, v.COLOR $\leftarrow$ WHITE
  $C = \phi$
  $flips = 0$
  k initialized by density
  while $flips < k$ **do**
    for Each white or grey sensor $v \in V$ **do**
      if Coin flip in range of $1 - (|v.Neighbors| + 1)^{-(\frac{1}{|v.Neighbors|})}$ **then**
        v.COLOR $\leftarrow$ BLACK C = $C \bigcup v$
      end if
    end for
  flips++
  end while

E. Random Distributed Dominating Set

Since the distributed with rounds algorithm could be implemented in a centralized or localized manner (centralized when an observer told sensors when to stop adding themselves, localized when k tosses were used) we next looked at a purely localized algorithm. Due to the randomized nature of the previous algorithm, k tosses would not necessarily create a CDS every time, although a CDS not being formed would be statistically unlikely. In this section, we present an algorithm for forming a dominating set (not connected). To compare this algorithm to the others in terms of size of CDS, we connect the components using phase 2 connection algorithm from GreedyConnect [5].

Each sensor tosses a coin weighted with the probability p as derived above. When the graph is sparse, a sensor is more likely to add itself to the dominating set than when the graph is dense. Any sensor which adds itself to the DS then alerts its neighbors to update their color to grey. Next, any remaining white sensor adds itself to the set. This ensures a dominating set because any non-dominated sensor will add itself and also dominate any white neighbors. This also limits the algorithm to two rounds. We assume that Phase 1 is completed in round 0 and Phase 2 in round 1. Using rounds to distinguish phases is a common technique in WSNs and these are usually implemented by using a fixed interval of time after a message signaled by the base station.

The number of messages passed is $O(|\text{DS}| * \Delta)$ since upon joining the DS a sensor alerts its neighbors, this is bounded by $O(n^2)$. Since sensors are doing the coin tosses asynchronously, the time complexity is constant.

Require: $\forall\ v \in V$, v.COLOR $\leftarrow$ WHITE
  $C = \phi$
  for Each sensor $v \in V$ **do**
    if Coin flip in range of $1 - (|v.Neighbors| + 1)^{-(\frac{1}{|v.Neighbors|})}$ **then**
      v.COLOR $\leftarrow$ BLACK
      for every neighbor $u \in N(v)$ **do**
        if u.COLOR == WHITE **then**
          u.COLOR $\leftarrow$GREY
        end if
      end for
    end if
  end for
  for Each sensor $v \in V$ **do**
    if v.COLOR == WHITE **then**
      v.COLOR $\leftarrow$ BLACK
      for every neighbor $u \in N(v)$ **do**
        if u.COLOR == WHITE **then**
          u.COLOR $\leftarrow$GREY
        end if
      end for
    end if
  end for

IV. Simulation Results

We compared the size of the CDS created by our three randomized algorithms with that of K2 [4] and GreedyConnect [5].

For each of our algorithms, our simulation takes in a randomly generated graph with a specified number of sensors distributed randomly across a 100 by 100 region and returns the size of the CDS after running a specific algorithm.

For the remainder of this section we discuss the results of running the simulation on each algorithm with the following conditions:

1) Five random graphs were created with sensors in each having the following ranges: 15, 20, 25, 30
2) Each graph consisted of 100 sensors distributed in a 100 by 100 region
3) Each algorithm was run on each graph five times

The algorithms we simulated would run asynchronously on actual sensor networks. For this reason we randomized the order of the sensors in our data structure and ran each algorithm multiple times on each graph. This simulates how even on the same graph, each algorithm may create a different CDS each time it is run.

A. Dominating Set Construction

The Random Distributed Dominating Set algorithm we discussed in Section III-E created a dominating set (not connected). In this section we compare the size of this DS with the greedy DS created by GreedyConnect [5] and presented in section II-B. In the following sections of this chapter we use the connection algorithm from Greedy Connect to connect the components and create a CDS.

In Figure 2 each point represents the average of the five runs of a particular algorithm on a particular graph. We see that the greedy DS is fairly consistent in size and that the randomized algorithm varies depending on the topology of the graph.

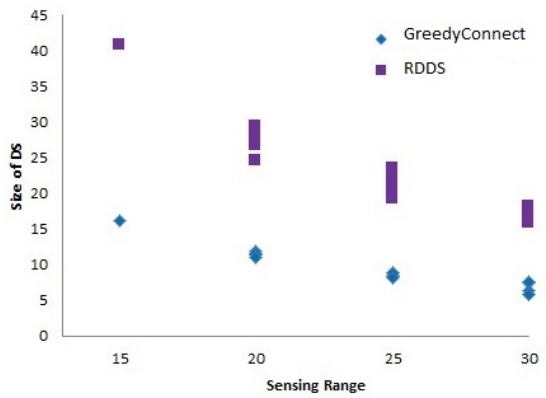

Fig. 2. Comparing the size of the DS for the Greedy and the Random Distributed Algorithm

Figure 3 contains the average DS size for a particular range. We note the greedy algorithm creates a DS 50% to 60% smaller than the size of the randomized algorithm for every range.

B. Connected Dominating Set Construction

In this section we compare the size of the CDS created by each algorithm. From Figure 4 we see that the size of the centralized randomized algorithm is not consistent based on range. This is because the results can vary significantly based on what order the nodes are picked in at random. The other four algorithms fairly consistently make the same size CDS based on sensing range. As can be seen from the figure, the Random Distributed algorithm presented in Section III-E performs the best and is pretty close to K2 even without any modifications, particularly at higher densities.

We next consider the average size of the CDS by sensing range as shown in Figure 5. The randomized

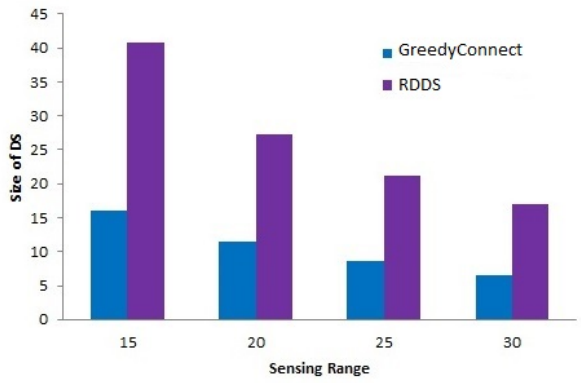

Fig. 3. Comparing the size of the DS for the Greedy and the Random Distributed Algorithm

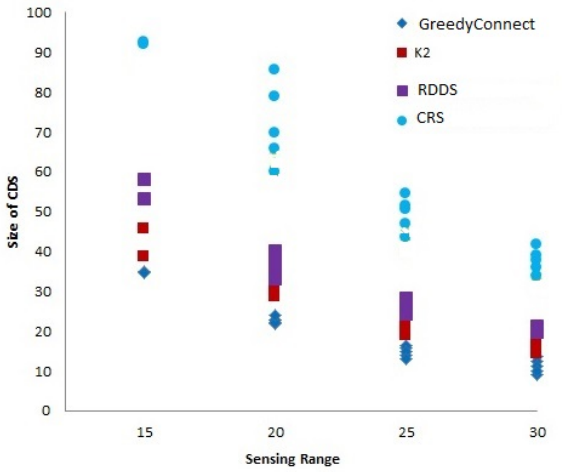

Fig. 4. Comparing the size of the CDS for each graph

CDS algorithms consistently performed worse than the other three algorithms and were particularly poor when the graphs were sparse. Greedy consistently out-performs every other algorithm and out-performs our reference algorithm, K2, by roughly 15%. The Random Distributed algorithm was about 10% percent worse than K2 but it should be noted that this can be a tradeoff for improved message and time complexity.

C. Trimming the CDS

The randomized algorithms were out-performed by both GreedyConnect and K2. However, all three randomized algorithms created their DS or CDS with less message passing and less time complexity. A CDS that can be formed very quickly would be useful, but not if it contains

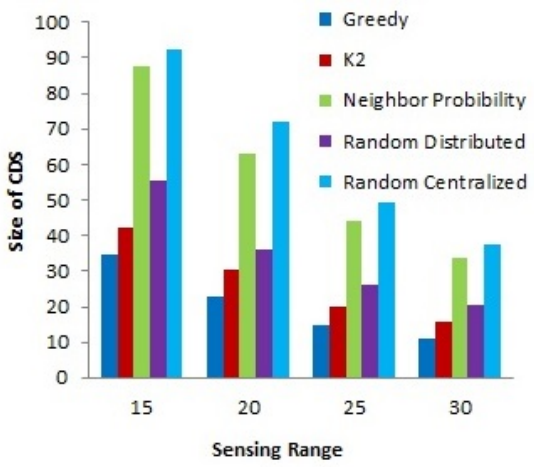

Fig. 5. Comparing the size of the CDS for the average of each range

most of the sensors in the WSN as was sometimes the case for our centralized algorithm.

For this reason, we ran the simulation again with the only change being the addition of the trimming algorithm from Phase 2 of K2 as explained in Section II-A. For every algorithm, after a CDS was found, the K2 trimming algorithm was run. This resulted in a dramatic drop in the size of the CDS for all three randomized algorithms. Comparing Figure 6 to Figure 4 the reduction in size for all three randomized algorithms is evident. Note that K2 does not change since its original implementation included both phases and no further reduction will result from repeating the trimming process.

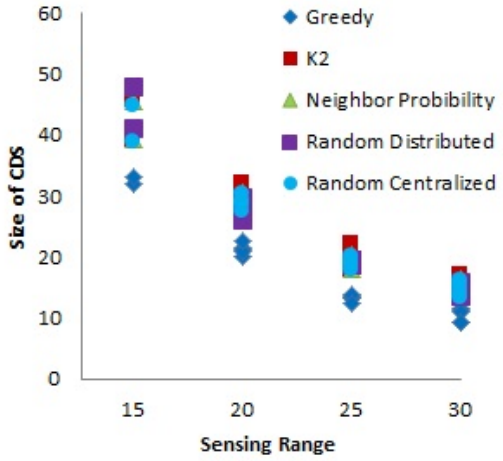

Fig. 6. Comparing the size of the CDS after trimming

We also compare the average CDS size by range after trimming in Figure 7. GreedyConnect still holds the lead for creating the smallest CDS by almost 25% for every range. However, all three randomized algorithms now out-perform K2 slightly except for range 15 where they are very close in size. We note that although the trim was run on GreedyConnect as well, the average size dropped very little. This can be seen in Table 8 which compares the size of the CDS before and after trimming. This is as expected since the GreedyConnect has fewer unnecessary vertices than the other algorithms. The randomized CDS algorithms were reduced in size by roughly 50% after trimming.

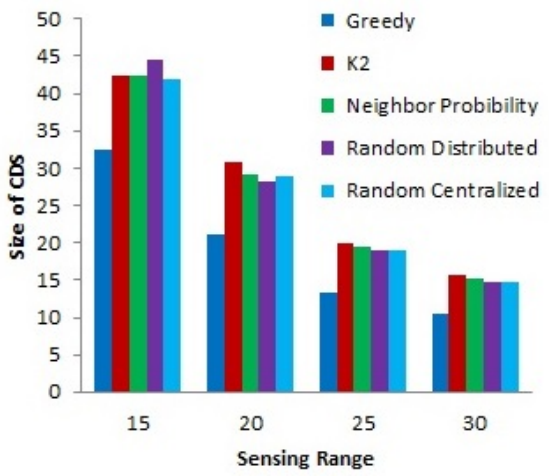

Fig. 7. Comparing the size of the CDS after trimming

	Origonal CDS Size				After Trimming			
	15	20	25	30	15	20	25	30
K2	42	31	20	16	42	31	20	16
Distributed Greedy	35	23	15	11	33	21	13	11
Random Centralized	92	71	49	38	42	29	19	15
Neighbor Probibility	88	62	44	34	43	29	20	15
Random Distributed	56	36	26	21	45	28	19	15

Fig. 8. Size of CDS before and after Trim

Next we summarize the time and message complexity of all the algorithms. As can be seen from the Table III all three randomized algorithms are better than both K2 and GreedyConnect in time and message complexity and make for excellent candidates to replace Phase 1 of K2. This lower complexity would translate into significant energy savings. Recall that the O(n^2) upper bound for the message complexity of Random Distributed a loose upper-bound and $O(|\text{DS}| * \Delta)$ is a more accurate representation.

V. Conclusion and Future Work

Using the CDS size as a metric, the GreedyConnect algorithm clearly out-performs K2 and the three new

TABLE III. Time and Message Complexity Comparison

Algorithm	Time	Message
Greedy Connect	$O(n^2)$	$O(n^2)$
K2 (Phase 1)	$O(\Delta^2)$	$O(\Delta)$
K2 (Phase 2)	$O(\Delta^2)$	$O(\Delta)$
Random Centralized	$O(n)$	$O(n)$
Neighbor Probability	$O(k)$	$O(n)$
Random Distributed	$O(1)$	$O(n^2)$

randomized algorithms. However, it sends a large number of messages. The results clearly show that the three algorithms presented exhibit a similar performance to K2 with lower time and message complexity. Since passing messages drains the battery, this may result in a significant improvement in the lifetime of the network. Further studies using implementation on a test bed are planned in order to quantify this gain.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications*, vol. 38, no. 4, pp. 102–114, 2002.

[2] C.-Y. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.

[3] B. Clark, C. Colbourn, and D. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, pp. 165–177, 1990.

[4] F. Dai and J. Wu, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 10, pp. 908–920, Oct. 2004. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2004.48

[5] A. Dhawan, M. Tanco, and N. Scoville, "A distributed greedy algorithm for constructing connected dominating sets in wireless sensor networks," in *SENSORNETS 2014 - Proceedings of the 3rd International Conference on Sensor Networks, Lisbon, Portugal, 7 - 9 January, 2014*, 2014, pp. 181–187.

[6] J. Yu, N. Wang, G. Wang, and D. Yu, "Connected dominating sets in wireless ad hoc and sensor networks–a comprehensive survey," *Computer Communications*, vol. 36, no. 2, pp. 121–134, 2013.

[7] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 56–73, 1987.

[8] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, Apr. 1998.

[9] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and K.-I. Ko, "A greedy approximation for minimum connected dominating sets," *Theor. Comput. Sci.*, vol. 329, no. 1-3, pp. 325–330, 2004.

[10] S. Funke, A. Kesselman, U. Meyer, and M. Segal, "A simple improved distributed algorithm for minimum cds in unit disk graphs," *ACM Trans. Sen. Netw.*, vol. 2, no. 3, pp. 444–453, Aug. 2006.

[11] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1597–1604.

[12] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-optimal connected dominating sets in mobile ad hoc networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. ACM, 2002, pp. 157–164.

[13] Y. Li, M. T. Thai, F. Wang, C.-W. Yi, P.-J. Wan, and D.-Z. Du, "On greedy construction of connected dominating sets in wireless networks: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 5, no. 8, pp. 927–932, Dec. 2005.

[14] D. Li, H. Du, P.-J. Wan, X. Gao, Z. Zhang, and W. Wu, "Construction of strongly connected dominating sets in asymmetric multihop wireless networks," *Theoretical Computer Science*, vol. 410, no. 810, pp. 661 – 669, 2009.

[15] M. T. Thai, R. Tiwari, and D.-Z. Du, "On construction of virtual backbone in wireless ad hoc networks with unidirectional links," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 9, pp. 1098–1109, 2008.

[16] F. Zou, Y. Wang, X.-H. Xu, X. Li, H. Du, P. Wan, and W. Wu, "New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs," *Theoretical Computer Science*, vol. 412, no. 3, pp. 198 – 208, 2011, combinatorial Optimization and Applications {COCOA} 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397509004162

[17] W. Duckworth and B. Mans, "Randomized greedy algorithms for finding small k-dominating sets of regular graphs," *Random Structures & Algorithms*, vol. 27, no. 3, pp. 401–412, 2005.

[18] G. Cao, "Distributed services for mobile ad hoc networks," Ph.D. dissertation, Citeseer, 2005.