A Distributed Self-Healing Approach for Virtual Backbone Construction and Maintenance in Wireless Sensor Networks

Najla Al-Nabhan, Mznah Al-Rodhaan and Abdullah Al-Dhelaan

Department of Computer Science, King Saud University, Riyadh, Saudi Arabia

Abstract: Future Wireless Sensor Networks (WSNs) will be composed of a large number of densely deployed sensors. Distributed techniques with fault-tolerance and energy awareness are expected in WSNs. Computing Connected Dominating Sets (CDSs) have been widely used for virtual backbone construction in WSNs to control topology, facilitate routing and extend network lifetime. This paper proposes a new distributed algorithm for CDS construction in WSNs. The algorithm is intended to construct a CDS with the smallest ratio when compared to its centralized version. Moreover, this paper proposes a localized algorithm that efficiently maintains the backbone when some backbone nodes decide to enter the energy saving sleep mode. We attempt to prolong the lifetime of the constructed CDS by allowing nodes with higher residual energy to have more chances to be part of the constructed and maintained backbone. Simulation shows that our distributed approach has a maximum ratio of 1.5 to the centralized approach and it satisfies all of the geometrical properties of its central version. Based on this ratio, this distributed algorithm has an approximation factor of 7.5 to the optimal CDS. To the best of our knowledge, this approximation outperforms the existing distributed CDS construction algorithms.

Key words: Approximation algorithms · Connected dominating set · Cooperative algorithms · Localized algorithms · Graph theory · Fault-tolerance

INTRODUCTION

A sensor network is a wireless network that consists of thousands of very small stations, called sensors. The main function of sensors is to monitor, record and notify a specific condition at various locations to other stations and end users. WSNs pose their unique challenges due to the lack of a central entity for organization, the sensors’ limitation and the mobility of participants, as well as the limited range of wireless communications. As each sensor node is tightly power-constrained and one-off, the lifetime of a WSN is limited. In order to prolong the network lifetime, energy-efficient protocols should be designed for the characteristic of WSNs. An important research problem in wireless-sensing networking is to find a small set of nodes that can collaborate to form a self-organizing network to substitute for the absence of infrastructure and central control. This virtual backbone plays a significant role in enhancing the network efficiency, extending its lifetime and supporting routing processes as well as all other network tasks and applications. For example, with a CDS, routing will be restricted to the reduced graph formed by the CDS. Every node in a CDS, called dominators, is considered as a gateway and only a gateway needs to keep routing information. Other nodes, called dominatees, need not keep any routing information. Moreover, any dominatee can switch to sleep mode for energy saving without causing network partition. In such a scenario, valuable transmission energy, nodes’ memory and computation time and bandwidth are saved. As backbone nodes are usually carrying higher load than other nodes (which are not in the CDS), they run out of their energy faster. One way to make network lifetime longer is to reduce the energy consumption of individual nodes by switching them back and forth between active and sleep modes. Hence, after a CDS is constructed, a self-healing technique is needed to be employed to maintain the backbone when a number (or a percentage) of backbone nodes decides to enter the energy-saving sleep mode [1, 2, 3]. Hence, after a CDS is constructed, an
efficient maintaining algorithm is needed to be employed
to maintain the backbone when a number (or a percentage) of backbone nodes decides to enter the energy-saving sleep mode.

The minimum connected dominating set (MCDS) problem is to find a connected dominating set with the smallest possible cardinality among all connected dominating sets on a graph $G$. The MCDS problem is defined as follows: For a given connected graph (network) $G=(V, E)$, where $V$ is the set of vertices (sensors) and $E$ is the set of edges that provides the available communications, a dominating set (DS) is a subset $V'$ of $V$, where for each vertex $u$ of $V$, $u$ is either in $V'$ or has at least one neighbor vertex in $V'$. The minimum dominating set (MDS) problem is to find a dominating set with smallest cardinality. The decision version of the MDS is a classical NP-complete problem. The dominating set $D$ is a connected dominating set if the sub-graph induced by vertices in $D$ is connected. Maximal Independent Set (MIS) is a subset $M$ of $V$ that satisfies the following conditions: i) nodes in the MIS are pairwise nonadjacent and ii) no more nodes can be added to maintain the non-adjacency property of this set. Each node that is not in the MIS is adjacent to at least one node in the MIS. If we connect all nodes in the MIS through some nodes not in the MIS, a CDS is then constructed [2].

In the context of CDS construction, the approximation ratio of an algorithm $A$ is defined as the largest (worst) ratio between the size of the obtained CDS using algorithm $A$ and the optimal result that can be obtained by MCDS (opt). CDS construction algorithms are usually classified into centralized [4, 5, 6] and distributed algorithms [7, 8, 9]. Centralized approaches usually achieve better performance than distributed approaches, but they utilize the global information of the network, so they are energy-consuming and hard to be realized in the practical applications. On the contrary, distributed approaches can be realized by sensors with low complexity and has become a hot topic. Additionally, WSNs are usually exposed to challenging and dynamic environments. Therefore, it is possible for connectivity loss of individual nodes to occur. In these situations, the conventional centralized algorithms that need to operate with global knowledge of the whole network will potentially experience a serious protocol failure as a result of transmission errors or a failure of a critical node [3]. Recently, a new centralized algorithm for constructing a CDS was proposed in [4, 5], with a constant approximation ratio of $5$. This approximation factor is the smallest in the literature. While most of the existing CDS construction algorithms are based on the conventional MIS that guarantees the distance between any complementary subsets is exactly two hops or at most three hops, the proposed approach in [4, 5] is based on constructing a special MIS which guarantees that the distance between any of its complementary subsets is exactly three hops. Unfortunately, the algorithm is centralized. Therefore, we need a distributed approach that offers energy awareness and backbone maintenance. In this paper, we propose a new distributed energy-aware algorithm for CDS construction and maintenance in wireless sensor networks. The proposed approach I extends the state-of-the-art centralized algorithm, proposed in [4, 5], in that it implements the exact-three-hop property in the distributed environment. Moreover, it employs a new ranking function that is carefully implemented to ensure that the constructed CDS is really connected. The introduced ranking function is intended to provide energy awareness. It attempts to prolong the lifetime of the constructed CDS by allowing nodes with higher residual energy for being more likely a part of the constructed backbone. In addition, the paper proposes a self-healing approach (approach II) to maintain the constructed backbone when a number (or percentage) of backbone nodes decides to enter the energy-saving sleep mode. The algorithm tries to locally fix the backbone without affecting the entire backbone. It limits the range of changes in the main backbone to at most three-hop distance from the sleeping node. We perform an extensive simulation study to evaluate the performance of the proposed methods over various parameters and settings. The rest of this paper is organized as follows: In section II, we present the related work. The rest of this paper is organized as follows: Next section presents the detailed design of the proposed approaches. Section 3 provides performance results and discussion. In section 4, we conclude our findings.

**MATERIALS AND METHODS**

In a distributed environment, the execution of algorithms is event-driven. Hence, the construction process of CDS is different to its centralized version in that it overlaps the construction phases. Based on the exchanged messages with its neighbors, a node is pre-programmed to change its color or trigger an event if a specific condition is satisfied. For illustrative purposes, we employ a coloring scheme to differentiate node states during the construction process. The nodes of $S_i$ (dominators) are marked black. The nodes used to cover
the disconnected regions ($S_1$ nodes) are marked red. Connectors are marked blue and dominatee nodes are marked gray. Other colors (white, orange and yellow) are temporarily introduced to make the elaboration of the algorithm easier: white is used for initialization, orange to mark nodes at a certain distance to a black node and yellow to mark disconnected components after $S_1$ construction and maintenance. As the information range of this distributed algorithm is at most three, each node records the important changes, specifically the color and region ID, in its 1-hop neighbors. Additionally, each node $i$ keeps the following three lists: 1) A \textit{Black List}: to store the IDs of its 1-, 2- and 3-hop black neighbors, as well as their corresponding graph distance to the node $i$; 2) A \textit{Red List}: to store the IDs of its 1-hop red neighbors; and 3) A \textit{Region List}: to store the region ID of its 1-hop neighbors. The described algorithm in the next subsection, \textit{approach I}, is a distributed CDS construction approach that implements the exact-three-hops property. Section 2.2 describes approach \textit{II}, which is a localized approach for CDS maintenance.

\textbf{Approach I: Distributed Algorithm for CDS Construction:} Approach I is a novel distributed approach for CDS construction based on the exact-three-hop model [4, 5]. In this model, a special independent set, called $S_1$, is first constructed. This set satisfies the following condition: The hop-distance between any two complementary subsets of $S_1$ is exactly three. Then, this model tries to obtain a small set of nodes to dominate the multiple disconnected regions (the yellow regions) resulted after constructing the special MIS in the first phase. Then, the obtained dominators in the first and second phases are connected to form the final CDS. However, in distributed setting, these phases are overlapped. The scenario of CDS construction in approach I is described as follows: given an arbitrary rooted spanning tree $T$, we define the tree level of a node $u$ as the number of hops in $T$ between $u$ itself and $i$, where $i$ is the root of $T$. All nodes are initially undominated and are colored white. The region ID of each node is set to its own ID. Nodes will be eventually marked with different colors during the execution of the algorithm and their region IDs might be changed.

The presented state diagram in Figure 1 shows the employed coloring mechanism to construct a CDS using approach I. First, the root node ($i$) initiates the CDS construction by coloring itself black. Then, $i$ broadcasts a “\textit{BLACK}” message that includes its ID and the counter value $k=3$. Upon receiving a “\textit{BLACK}” message with $k=3$, a white node colors itself gray, updates its black list, decrements $k$ by 1 and rebroadcasts the message with the decremented $k$ value; upon receiving a “\textit{BLACK}” message with $k=2$, a white node marks itself yellow, updates its black list, decrements $k$ by 1 and rebroadcasts the message with the decremented $k$ value; and upon receiving a “\textit{BLACK}” message with $k=1$, a white node marks itself orange and updates its black list. For the completion of this distributed algorithm, we declare the following events:

\textbf{Event: Orange Bidding:} Orange bidding is done by orange nodes and handled (evaluated and acknowledged) by their 1-hop yellow neighbors. The winning orange node is colored \textit{black}. After it is colored orange, each orange node announces its bidding by broadcasting an \textit{orange bidding message} that includes: orange node ID, level, residual energy value and number of its 3-hop black neighbors. Then, it waits for the acknowledgment from all of its yellow neighbors. Upon receiving an orange bidding from an orange node by one of its neighboring yellow nodes, the receiving yellow node checks the received bidding from all of its 1-hop orange neighbors, selects the orange node that has the lowest level, higher residual energy and/or maximum number of black neighbors and then it acknowledges the winning orange node by broadcasting an \textit{ACK message} with the winning node ID. Upon receiving \textit{ACK messages} from all its yellow neighbors, an orange node colors itself black and broadcasts a “\textit{BLACK}” message with its ID and $k=3$. Similar to the previous processing of black messages, upon receiving a “\textit{BLACK}” message with $k=3$, a white/orange/yellow node marks itself gray, updates its...
black list, decrements \( k \) by 1 and rebroadcasts the message with the new \( k \) value. Upon receiving a “BLACK” message with \( k = 2 \), a white/orange node marks itself yellow, updates its black list, decrements \( k \) by 1 and rebroadcasts the message with new \( k \) value. Upon receiving a “BLACK” message with \( k = 1 \), a white node marks itself orange, updates its black list and announces its bidding. Upon receiving a “BLACK” message with \( k = 1 \), an orange node updates its black list and announces its new bidding.

**Event: Yellow Bidding:** Yellow bidding is done and handled by yellow nodes. The winning yellow node is colored red. Each yellow node that has no orange/white neighbors announces its bidding by broadcasting a yellow bidding message. The yellow bidding message includes: Its ID, the number of its 1-hop yellow neighbors (called coverage factor) and residual energy value. Then it waits for the acknowledgment from all of its 1-hop yellow neighbors. A yellow node that has no yellow/orange/white neighbors will color itself red without bidding. Upon receiving a yellow bidding message by a yellow neighbor, it evaluates and acknowledges the highest bidding (the yellow node that has the highest coverage factor and/or higher residual energy) by broadcasting an ACK message with the winning node ID. Upon receiving ACK messages from all its 1-hop yellow neighbors, a yellow node colors itself red and broadcasts a “RED” message. Upon receiving the “RED” message by a yellow node, it colors itself gray and broadcasts an “YGRAY” message for its 1-hop neighbors. Upon receiving an “YGRAY” message, a yellow node recalculates and announces its new yellow bidding. The “YGRAY” message is very important for the execution of the algorithm in the distributed setting. It notifies yellow neighbors to: i) recalculate their bidding and ii) to reevaluate and confirm the previously received bidding as required.

**Event: Selecting Red Connectors:** The distributed logic yields red nodes having the same geometrical properties of the red node in the centralized algorithm: each red node has at least one black node that is only two hops away. Therefore, we need to include at most one gray node to connect each red node to its nearest black node. After a node is colored red, it selects a connector of its 1-hop gray neighbors based on number of their 1-hop backbone nodes. As a result, the winning gray node is colored blue and its 1-hop backbone nodes (black, red and blue) are merged into one region. If a red node already has a blue node within its 1-hop neighbors, it does not need to introduce a new connector. Instead, it links itself to that existing connector by sending a request-to-connect message and changes its region ID accordingly.

**Event: Changing the Region ID:** In this implementation, all nodes with the same region ID form a connected component. When a node changes its region ID, it notifies its 1-hop neighbors for this update by sending a region-change message. Upon receiving a region-change message by a region member (a backbone node that belongs to that region), it changes its region ID and broadcasts a region-change message for its 1-hop neighbors.

**Event: Gray Bidding:** Gray bidding is done by gray nodes and handled by their gray, blue, black and red neighbors. As a result, the winning gray nodes are colored blue and their 1-hop backbone nodes (black, red and blue) are merged into one region. Each gray node that has no yellow/orange neighbors and has more than one nearby region announces its gray bidding by broadcasting a gray bidding message that includes: its ID, its dominator color and region ID, count and a list of its 1-hop different regions. Upon receiving a gray bidding message by a backbone node (blue, red and black), it decides if it needs to connect to the reported regions by comparing its current region ID with the received region list. If different regions are going to be merged through this gray node, the receiving backbone node sends an ACK message to the bidding gray node in order to color it blue. As a result, the acknowledged gray node colors itself blue, calculates the new region ID and announces it by broadcasting a “BLUE” message, to its 1-hop backbone nodes in order to merge them all into one region. Upon receiving a gray bidding message by a neighboring gray node, it checks whether or not their dominators are black and have different region IDs. If yes, the receiving gray node sends an ACK message to the bidding gray node. The biding gray nodes are evaluated by their neighboring backbone nodes based on their exchanges region list. Upon receiving an ACK message from a neighboring gray node, the receiving gray node colors itself blue, calculates the new region ID and broadcasts a “BLUE” message with the new region ID. Upon receiving a “BLUE” message by a neighboring backbone node, a backbone node changes its region ID and broadcasts a region change message for its 1-hop neighbors. Upon receiving a “BLUE” message by a neighboring gray node, it updates its region list. The final CDS is obtained after executing all the triggered operations.
events. At that time, all white, orange and yellow nodes are colored black, red, blue, or gray. The union of the black, red and blue sets forms the final CDS.

Figure 1 shows the state diagram for the color changes within the execution of approach I. The transition conditions are described as follow: (a) is receiving a “BLACK” message with \( k=3 \), (b) is receiving a “BLACK” message with \( k=2 \), (c) is receiving a “BLACK” message with \( k=1 \), (d) is receiving ACK to orange bidding from all yellow neighbors, (e) is receiving ACK to yellow bidding from all yellow neighbors, (f) is receiving a “RED” message, (g) is receiving a “Request-to-connect” message and (h) is receiving ACK to gray bidding. Figure 2 shows an exemplary graph for a network of 100 nodes, deployed in a 100\( m \times 100m \) square field, after constructing a CDS using approach I. The transmission range of nodes is uniform and assumed to be 20\( m \). The black, red, blue nodes with edges between them form the backbone. The other gray nodes are the dominatees. The gray nodes are dominated by their black or red neighbors. The blue nodes are used to connect the red and black nodes to form the final backbone.

**Approach II: A localized Self-Healing Approach for CDS Maintenance:** This section discusses the design of approach II that efficiently maintains the backbone when some backbone nodes either run out of their energy or decide to enter the energy saving sleep mode. It tries to locally fix the backbone without affecting the entire backbone. To select the sleeping nodes, the program randomly selects a set of backbone nodes to sleep, or a backbone node decides to enter the sleep mode if a previously specified condition is satisfied (e.g. energy threshold). Before it decides to enter the sleep mode, a dominator node \( i \) initiates a preparation phase by broadcasting a SLEEP message to notify its dominatees. Upon receiving a SLEEP message by one of \( i \)’s dominatees \( d \), \( d \) checks its 1-hop neighbors to find another backbone node (a dominator or a connector) with highest residual energy. If an active backbone node exists within \( d \)’s 1-hop neighbors, \( d \) sends a request-to-dominate message to that backbone node. Upon receiving a request-to-dominate message by a backbone node, it adds \( d \) to its dominatees’ list and sends ACK message to \( d \). Upon receiving ACK message, \( d \) announces its new dominator by broadcasting a CONNECTED message with the ID of its new dominator. Upon receiving a CONNECTED message, \( i \) removes \( d \) from its dominatees’ list. If \( d \) does not have any active backbone node within its 1-hop neighbors, \( d \) calculates its disconnected gray list which contains the IDs of all its gray neighbors those are currently dominated by \( i \) and did not send a CONNECTED message. Then, \( d \) sends a DISCONNECTED message to \( i \) which includes \( d \)’s gray list and its energy level. Upon receiving DISCONNECTED message, \( i \) adds \( d \) to the disconnected list. Upon receiving a CONNECTED/DISCONNECTED message from all dominatees, \( i \) recalculates the gray coverage of the disconnected dominatees as number of their disconnected 1-hop neighbors. If disconnected list contains one disconnected node \( d \), then \( i \) selects \( d \) to be a compensatory dominator by sending a DOMINATOR message to it. If the disconnected list contains more than one disconnected node, \( i \) checks the disconnected list to find the minimal set of compensatory dominators. The disconnected node \( d \) that has the maximum gray coverage and the maximum energy level is selected by \( i \) to be a compensatory dominator. Upon receiving a DOMINATOR message by \( d \), it colors itself green, changes its dominator ID to its own ID and broadcasts a “GREEN” message. Upon receiving \( d \)’s GREEN message by \( i \), it adds \( d \) to the compensatory dominating list. Upon receiving \( d \)’s “GREEN” message by a disconnected gray neighbor, it sets the id of its dominator to the \( d \)’s ID and broadcasts a CONNECTED message. Upon receiving a “GREEN” message by a neighbor node, it updates its current neighboring dominator list. Then, \( i \) sends a DOMINATOR message to \( d \) and repeats the selection step of a new compensatory node until the disconnected list becomes empty. At that time, \( i \) announces to its 1-hop neighbors the compensatory dominating list which contains the introduced dominators as well as the old connectors of node \( i \). Furthermore, \( i \) selects the left most node \( l \) in the compensatory dominating set to initiate connecting all nodes in the compensatory dominating list together with
the old connectors of \(i\). The objective of this step is to find a path that ensures the connectivity of network parts after \(i\) is turned off. The role of the node \(i\) completes then.

Given a compensatory dominating list to the selected initiator \(l\), \(l\) first marks itself connected and adds all its 1-hop neighbors to the added list and broadcasts an ADD message. Upon receiving an ADD message, a node changes its state to added, broadcasts ADDDED message with a list of its compensatory neighbors that are not yet added. If some nodes in the compensatory dominating set are still un-added (or disconnected), then \(l\) selects its right most node \(r\) that has maximum number of disconnected compensatory dominators to be a new connector and sends it a CONNECT message. Upon receiving a CONNECT message, \(r\) changes its color to blue, sends an ACK message to \(l\) and broadcasts an ADD message. Upon receiving ACK message by \(l\), it adds \(r\)'s neighbors to the added list. Similarly, upon receiving an ADD message by another disconnected compensatory node, it marks itself added, sends ACK to the node \(r\) that added it and reports its disconnected compensatory neighbors to it. The algorithm repeats until all nodes in the compensatory list are connected.

RESULTS AND DISCUSSION

In this section, we discuss the performance of our proposed approaches and report simulation results. We first discuss the approximation factor and analyze the message and time complexities of our proposed approaches. Then, we describe the design of our simulation experiment; including environment setting, simulation input, deployment models and energy model. Finally, we show simulation results, with each representing an average of 50 runs.

Performance Discussion: For the approximation factor of approach I, compared to the proposed centralized algorithm in [4][5], the constructed \(S_1\) and \(S_2\) sets in approach I form independent sets in the network and their size is bounded by the size of the MCDS (1 opt). The \(S_2\) connectors are chosen by \(S_1\) nodes and are also bounded by the size of the \(S_1\) set, which is 1 opt. For \(S_1\) connectors, these connectors are 2-hop connectors. Unlike the centralized algorithm, the upper bound of these connectors is difficult to predict in the distributed environment. As a result of making decisions distributed and the lack of information, some redundant \(S_1\) connectors might be introduced by the algorithm. However, the implemented ranking function tries to minimize this duplication. Under different settings, simulation shows that approach I generates a CDS of a size bounded by 1.5 times than the size of the constructed CDS by its centralized algorithm in [4, 5]. As the approximation factor of this centralized algorithm is bounded by 5 opt [4, 5], these extensive experimental results establish a 7.5 opt as an upper bound for the size of the constructed CDS by approach I. Details on experimental results are presented in the next subsection. The information range shows the amount of neighborhood information that should be collected by a node to perform the CDS construction task. This factor has an influence on algorithm and message complexities. The Information range of approaches I and II is three. For time and message complexities, after a rooted spanning tree \(T\) is constructed, the construction of \(S_1\) and \(S_2\) dominators and their connectors uses linear messages and takes at most linear time. The construction of the spanning tree \(T\) depends on the employed algorithm. Spanning tree algorithms are expected to use linear messages and take either a linear or a linearithmic time. Thus, besides the construction of the tree \(T\), approach I uses O(n) messages and takes O(n) time. Approach II also uses at most O(n) messages and takes at most O(n) time.

Simulation Setup: In order to compare the size of the generated CDS by the distributed approach (approach I) to the centralized approach that extends it [4, 5], we apply both approaches to the same network topology (the same input). In addition, we applied approach II to maintain the generated CDS. For each reported simulation result, we generated 50 different network topologies. We investigated the performance of all algorithms with different input values for the number of nodes \(N\), the transmission range \(R\) and the deployment area \(A\). For a given 2D square field of area \(A\), we generate a total of \(N\) nodes. In our simulation, we mainly consider the uniform random deployment model, where each of the \(N\) sensors has equal probability of being placed at any point inside the given deployment field [10]. For the energy model, in order to calculate the total dissipated energy by our distributed approaches, we assigned the amount of energy dissipated by transmitter and receiver electronics to process a bit \(E_{elec}\) to be 50 nj/bit, the energy dissipated by transmitter amplifier to transmit a bit \(E_{amp}\) to be 0.1 nj/bit and the maximum size for the control packets to be 64 bits. The energy dissipated to receive \(k\) bits is calculated as: Energy_{receive}(k) = (E_{elec} \times k) [3]. The dissipated energy to transmit \(k\) bits for a distance \(d\) is calculated as: Energy_{transmit}(k, d) = (E_{elec} \times k) + E_{elec} \times k \times d^2 [3]. We studied
the performance with different input values for the numbers of nodes $N \in [100–1000]$ nodes, the transmission range $R \in [10–50]m$ and the deployment area $A \in [40×40–100×100]m^2$. We used UDG to model the network. Hence, the transmission range for all nodes is unified and equals $R$. We consider $R$ values that keep the network connected. For the deployment area, the choice of different field sizes for the same input size allows the generation of relatively sparse (for larger squares) and dense (for smaller squares) graphs [11]. For a set of $N$ nodes deployed in a field of area $A$, with each node having a transmission range $R$, we define a constant node density $\rho = \frac{N}{A}$, which denotes the expected number of nodes per unit area. We also define the expected number of nodes per transmission unit (or average node degree) $\gamma = \frac{\pi R \rho}{2}$, where $\rho$ is the expected node density [12].

Simulation Results: Simulation shows that the implementation of our distributed algorithm performs appropriately in constructing CDS for all network types and densities. It also shows that it satisfies the exact-three-hops property between each black node and its nearest black node. Moreover, red nodes are originally yellow nodes that are marked yellow using one of their 2-hop black neighbors (after receiving a “BLACK” message with $k=2$). Therefore, this implementation ensures that all red nodes are exactly 2-hop from at least one black node. This implementation also satisfies that the count of red connectors is at most equal to the count of red nodes and the count of black connectors is at most twice the count of black nodes. Figure 3 (a) shows the average CDS size in the distributed and centralized approaches when the number of nodes $N$ varies between 100 and 1000 nodes, $R=20m$ and $A=100×100m^2$. It also shows the average size of the maintained backbone after sleeping 5%, 25% and 50% of the original CDS nodes. For the distributed construction, obviously, the distributed construction produces a CDS of larger size than the centralized algorithm because decisions are not made based on the global view of the network. This increase in CDS size is almost a result of having a larger number of black connectors than in the centralized version. For the distributed maintenance, the figure shows that the size of the maintained backbone is slightly increases. By choosing the compensatory nodes from the existing connectors and dominatess of the sleeping node we allow limiting the changes in the maintained backbone to at most three-hop distance from the sleeping node. Figure 3 (b) shows the ratio of the distributed construction to the centralized construction and the ratio of the maintained backbone using our proposed self-healing approach to the constructed CDS using our distributed approach after sleeping 25%, 50% and 95% of the backbone nodes.

For energy efficiency and communication complexity, Figure 4 (a) shows the total dissipated energy in micro joule ($\mu J$) per number of nodes (the average energy dissipation per node) in approach I. It is shown that energy consumption increases while the number of nodes increases due to the increase of nodes’ degrees. Energy consumption is proportional to the number of exchanged messages. Therefore, the increase in nodes’ degree increases the number of transferred/received control messages (color announcements, ACK, bidding) by nodes. Each arrival/transmission of a message requires a certain amount of energy to be dissipated by the receiving/transmitting node. For message complexity, the corresponding average message complexity per number of nodes to above results is shown in Figure 4 (b). The corresponding average node per unit of area (node degree) for the presented variation in number of nodes from 100 to 1000 varies between 12.6 nodes and 125.6 nodes. Comparing the average number of transmitted messages per node to the average node degree in different setting, we identify that the average number of transmitted messages per node is bounded by the average node degree for the different variations of network setting. This comparison is based on the average number of exchanged messages by the algorithm and the average node degree. The average node degree is bounded by the number of nodes in the network ($N$). Therefore, our analysis for the message complexity of approach I as $O(n)$ is correct.

For varying the transmission range, Figure 5 shows the impact of changing the transmission range $R$ from 10m to 50m, when $N=400$ and $A=100m×100m$. Each time we increment $R$ value, we move from sparse network toward dense network. The CDS size decreases each time we increase $R$ value. The transmission range is inversely proportional to the size of CDS. The increase in transmission range allows a dominator node to dominate farther nodes. It also allows a connector node to connect many dominators at the same time. Hence, the CDS size is reduced as a result of having fewer dominators and connectors. This relationship is valid for any CDS construction algorithm. For varying the deployment area, for $N=400$ nodes and $R=10m$, we changed the size of the deployment field between $40m×40m$ and $100m×100m$. The choice of different field sizes for the same $N$ and $R$ allows generating a relatively sparse (for larger squares) and dense networks (for smaller squares).
Fig. 3: The average performance of approaches I and II vs. the centralized approach when N varies from 100 to 1000 nodes. (a) Shows the average CDS size. (b) Shows the ratio

Fig. 4: Energy efficiency and communication overhead of approaches I: (a) The average energy dissipation per node in (μJ). (b) The average number of transmitted messages per node
In this particular example, the corresponding average number of nodes per unit area is 78.5, 34.8, 19.6 and 12.5 for the field sizes, respectively. Results are shown in Figure 6.

CONCLUSIONS

In this paper, we propose a distributed energy-aware approach for CDS construction and maintenance in wireless sensor networks. Approach I aims to construct a small CDS in distributed setting with energy awareness. It extends the exact-three-hops property that was introduced in the centralized algorithm in [4, 5] to the distributed environment. Approach II aims to achieve efficient CDS maintenance. Throughout the simulation, we study the performance of our distributed construction approach and compare it to the centralized construction. Simulation shows that our distributed construction approach has a maximum ratio of 1.5 to the centralized approach. Based on this ratio, this distributed algorithm has an approximation factor of 7.5 to the optimal CDS. This approximation is the smallest among all existing distributed CDS construction algorithms. Simulation also shows that our distributed self-healing approach doesn’t cause high stretches to the maintained backbone. In future, we consider applying a pruning algorithm to remove the redundant connectors from the final backbone. Furthermore, we plan to consider another source for heterogeneity by assuming sensor nodes to have variable transmission ranges. Moreover, we consider implementing an adaptive approach that allows and reacts to a limited mobility of CDS nodes and non-CDS nodes.

ACKNOWLEDGMENTS

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group no. RGP-VPP-264.

REFERENCES