A Distributed Fault-Tolerant Topology Control Algorithm for Heterogeneous Wireless Sensor Networks

Hakki Bagci, Ibrahim Korpeoglu, *Senior Member, IEEE*, and Adnan Yazıcı, *Senior Member, IEEE*

Abstract—This paper introduces a distributed fault-tolerant topology control algorithm, called the Disjoint Path Vector (DPV), for heterogeneous wireless sensor networks composed of a large number of sensor nodes with limited energy and computing capability and several supernodes with unlimited energy resources. The DPV algorithm addresses the k-degree Anycast Topology Control problem where the main objective is to assign each sensor's transmission range such that each has at least k-vertex-disjoint paths to supernodes and the total power consumption is minimum. The resulting topologies are tolerant to $k-1$ node failures in the worst case. We prove the correctness of our approach by showing that topologies generated by DPV are guaranteed to satisfy k-vertex supernode connectivity. Our simulations show that the DPV algorithm achieves up to 4-fold reduction in total transmission power required in the network and 2-fold reduction in maximum transmission power required in a node compared to existing solutions.

Index Terms—Topology control, fault tolerance, k-connectivity, disjoint paths, heterogeneous wireless sensor networks, energy efficiency

1 Introduction

Wireless sensor networks (WSNs) have been studied extensively for their broad range of potential monitoring and tracking applications, including environmental monitoring, battlefield surveillance, health care solutions, traffic tracking, smart home systems and many others [1]. WSNs are typically composed of large number of tiny sensors that are capable of sensing, processing and transmitting data via wireless links. Sensor nodes collaborate in a distributed and autonomous manner to accomplish a certain task, usually in an environment with no infrastructure.

Power efficiency and fault tolerance are essential properties to have for WSNs in order to keep the network functioning properly in case of energy depletion, hardware failures, communication link errors, or adverse environmental conditions, events that are likely to occur quite frequently in WSNs [2], [3]. Topology control is one of the most important techniques used for reducing energy consumption and maintaining network connectivity [4]. There are many reactive and proactive topology control techniques for tolerating node failures in WSNs [37].

In this paper, we focus on a proactive fault-tolerant topology control algorithm in heterogeneous WSNs with a two-layered architecture where the lower layer consists of low-cost ordinary sensor nodes, with limited battery power and short transmission range. The upper layer consists of supernodes, which have more power reserves and better processing and storage capabilities. Links between supernodes have longer ranges and higher data rates; however, supernodes are fewer in number due to their higher cost. Supernodes can also have some special abilities like acting against an event or a certain condition. This type of supernodes are called actors (or actuators), and sensor networks that contain actors are called wireless sensor and actor networks (WSAN). In WSANs, data gathered by sensors is forwarded to actors for performing the required actions [5]. A heterogeneous WSN with supernodes are known to be more reliable and have longer network lifetime than the homogeneous counterparts without supernodes. Heterogeneity can triple the average delivery rate and provide a five-fold increase in the network lifetime if supernodes are deployed carefully [6].

This paper introduces a new algorithm called the Disjoint Path Vector (DPV) algorithm for constructing a fault-tolerant topology to route data collected by sensor nodes to supernodes. In WSNs, guaranteeing k-connectivity of the communication graph is fundamental to obtain a certain degree of fault tolerance [4]. The resulting topology is tolerant up to $k-1$ node failures in the worst case. We propose a distributed algorithm, namely the DPV algorithm, for solving this problem in an efficient way in terms of total transmission power of the resulting topologies, maximum transmission power assigned to sensor nodes, and total number of control message transmissions. Our simulation results show that our DPV algorithm achieves between 2.5-fold and 4-fold reduction in total transmission power required in the network, depending on the packet loss rate, and a 2-fold reduction in maximum transmission power required in a node compared to existing solutions. The power efficiency of our algorithm directly results from the novel approach that we apply while discovering the disjoint paths. This approach involves in storing full path information instead

- *H. Bagci and A. Yazıcı are with the Department of Computer Engineering, Middle East Technical University, Ankara, Turkey. E-mail: hakki.bagci@tubitak.gov.tr.*
- *I. Korpeoglu is with the Department of Computer Engineering, Bilkent University, Ankara, Turkey.*

Manuscript received 10 Nov. 2013; revised 8 Feb. 2014; accepted 12 Mar. 2014. Date of publication 8 Apr. 2014; date of current version 6 Mar. 2015. Recommended for acceptance by D. Manivannan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2014.2316142

of just next node information on the paths and provides a large search scope for discovering the best paths throughout the network without the need of global network topology.

The paper is organized as follows: In Section 2 we present related work on topology control algorithms for WSNs. We describe our approach and our DPV algorithm in Section 3 and present our simulation results in Section 4. We conclude our paper in Section 5. We also include a supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2014.2316142, providing further examples and details about the algorithm.

2 Related Work

Topology control is defined as controlling the neighbor set of nodes in a WSN by adjusting transmission range and/or selecting specific nodes to forward the messages [7]. Topology control approaches can be divided into two main categories, namely, homogeneous and nonhomogeneous [4]. In homogeneous approaches transmission range of all sensors are the same whereas in nonhomogeneous approaches nodes can have different transmission ranges.

There are many topology control methods proposed in literature and they can be classified according to the techniques they use [34]. Many topology control methods [8], [9], [10], [11], [12], [13] are built on the transmit power adjustment technique which depends on the ability of sensors to control their transmit power. Some algorithms [15], [14], [16], [17], [18] use sleep scheduling which aims to decrease energy consumption while nodes are in idle state. Others use geometrical structures, location and direction information and also combinations of these techniques [19], [20], [21], [22], [23], [24], [25] and [36]. The difference between these studies and our work is that we try to minimize nodes' total transmission power in two-tiered heterogeneous topologies whereas other works focus on flat homogeneous topologies. In addition we focus on connectivity between a sensor node and supernodes whereas they focus on connectivity between any two nodes.

Clustering can also be considered as another way of topology control, where the aim is to organize the network into a connected hierarchy for the purpose of balancing load among the nodes and prolonging the network lifetime [26]. Hierarchical clustering techniques [27], [28] and [29] select cluster heads depending on various criteria and create a layered architecture. However, these techniques start with a flat topology and end up with a layered one. On the other hand, we start with a layered architecture from the beginning, where the supernodes are already given. Instead of building clusters, we focus on maintaining fault-tolerant connectivity between sensor nodes and supernodes.

An active research area where layered and heterogeneous architectures are utilized is wireless sensor and actor networks. WSANs usually have a two-layer architecture where the lower level is composed of low cost sensor nodes and the upper layer consists of resource-rich actor nodes which take decisions and perform appropriate actions [5]. In WSANs, there are usually two type of wireless communication links: actor-actor and sensor-actor links [30]. The links between sensors and actors are assumed to be less reliable [31], hence there are several methods proposed for maintaining reliable sensor-actor connectivity [30], [31], [32]. The methods of [31] and [32], however, do not employ k-connectivity between sensors and actors and thus they do not guarantee fault-tolerance in case of $k-1$ node failures. Although [30] addresses the k-actor connectivity problem, it does not consider the energy efficiency of the resulting topologies. Our approach differs from these works by maintaining k-connectivity and addressing power efficiency at the same time.

A prominent work on fault-tolerant topology control for heterogeneous WSNs with a two-layer network architecture is proposed by Cardei et al. [35], addressing both k-connectivity and energy efficiency. As we do, they focus on the k-degree Anycast Topology Control (k-ATC) problem, which aims adjusting the transmission range of the sensor nodes to achieve k-vertex supernode connectivity and minimize the nodes' maximum transmission power. They propose a greedy centralized algorithm called global anycast topology control (GATC), and also a distributed algorithm called distributed anycast topology control (DATC), which provides k-vertex supernode connectivity by incrementally adjusting the transmission range of the sensor nodes. GATC is mostly of theoretical importance since it is not practical to apply it for large scale WSNs due to the requirement of global topology knowledge. The DATC algorithm [35] is a distributed and hence a more practical solution to the k-ATC problem. This algorithm requires only 1-hop neighborhood topology information, which can also be extended to h-hop. The objective of DATC is to ensure that any neighbor node u, in the reachable neighborhood of any node v, is either directly reachable from node v or there are at least k-vertex disjoint paths from v to u.

Our algorithm differs from DATC by the approach that we adopt for discovering vertex disjoint paths. In DATC, each node starts with a minimal set of neighbors and minimal power level. The power level is increased incrementally and only the paths from the neighborhood that is reachable with that power level can be discovered. The nodes outside of the reachable neighborhood are totally unknown to the node performing discovery and thus they are out of the search scope for discovering paths. This is an important limitation for DATC because it has a low chance to find k-vertex disjoint paths for its neighbors in its reachable neighborhood, which typically has nodes that are in 1 or 2 hops distance from the node performing the discovery. In contrary to DATC, in our algorithm, a sensor node can discover paths including nodes outside of its reachable neighborhood. This is achieved by storing full path information from supernodes to sensor nodes in local information tables. In this way, the DPV algorithm has more chance to discover better k-disjoint paths than DATC. Another difference of our algorithm from DATC is that, we decrease the power level only after deciding the final topology. During path discovery in the DPV algorithm, nodes operate with maximum power, thus, increasing the likelihood of discovering more paths than DATC. Our simulation results are in conformity with this discussion.

3 Disjoint Path Vector Algorithm

We propose a new algorithm, called Disjoint Path Vector Algorithm, for fault-tolerant topology control in two-tiered heterogeneous WSNs consisting of resource-rich supernodes and simple sensor nodes with batteries of limited capacity. Before introducing our algorithm we give some necessary definitions:

Definition 1 (Vertex disjoint paths). *A set of paths with common end points that have no other vertices in common.*

Definition 2 (k-vertex supernode connectivity). *A WSN is k-vertex supernode connected if, for any sensor node $v \in V$, there are k pairwise vertex disjoint paths from v to one or more supernodes. In other words, a WSN is k-vertex supernode connected if the removal of any $k-1$ sensor nodes does not partition the network [35].*

To obtain fault-tolerant topologies we focus on k-vertex supernode connectivity where each sensor is connected to at least one supernode by k-vertex disjoint paths. Such topologies are tolerant to $k-1$ node failures in the worst case. Our algorithm is based on the observation that for a node we can remove the edges with neighbors that are not on one of the k-vertex disjoint paths from the node to one of the supernodes. To achieve this, we need to determine which neighbors are on one of such paths and which are not. Our DPV algorithm finds a superset of the required vertices in order to guarantee the k-vertex supernode connectivity. Having found the required neighbors, each node removes edges not connected to a required node in coordination with its neighbors. Then, to save energy, we decrease the transmission range of the sensor nodes but still reach the farthest node in the new set of neighbors.

DPV is a distributed algorithm executed by each sensor node in the network. Each node uses topology information in its 1-hop neighborhood. Paths to the supernodes are explored using messages, which contain path information from a supernode to a sensor node. Global network topology information is not required by the DPV algorithm.

3.1 Network Model

We consider a heterogeneous WSN consisting of M supernodes and N sensor nodes, with $M \ll N$. Sensor nodes are randomly deployed in the 2D plane. Supernodes are deployed at known locations manually. We are interested in sensor-sensor and sensor-supernode communications only. We do not model supernode-to-supernode communications because we assume that supernodes are not energy constrained and thus can directly communicate with a base station or can send data collected from sensors to other supernodes if necessary. Delivering a message originated at a sensor node to any of the supernodes is considered a successful delivery. In the initial network topology each sensor node has transmission range R_{max}. We represent the initial network topology with an undirected weighted graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N, v_{N+1}, \ldots, v_{N+M}\}$ is the set of nodes and $E = \{(v_i, v_j) \mid dist(v_i, v_j) < R_{max}\}$ is the set of edges, where $dist(v_i, v_j)$ depicts the distance between nodes v_i and v_j. The first N nodes in V are the energy-constrained sensor nodes and the last M nodes are the resource-rich supernodes.

3.2 Problem Statement

We aim to construct a k-vertex supernode connected network topology to route data collected by sensor nodes to the supernodes for two-tiered WSNs with the network model given in Section 3.1. We model topology control as a transmission range assignment problem for each sensor node in the network. The objective is to minimize the assigned transmission power for all sensors while maintaining k-vertex disjoint paths from each sensor to the set of supernodes. In this topology, each sensor node in the network must be connected to at least one supernode with k-vertex disjoint paths.

We can define the problem as follows: given a k-vertex supernode connected WSN with M supernodes and N energy-constrained sensor nodes that can adjust their transmission range up to a predefined constant R_{max}, determine the transmission range of each sensor such that the total transmission power is minimized and the resulting topology is still k-vertex supernode connected. Next, we state the problem definition more formally.

Problem Definition: Given an undirected graph $G = (V, E)$, where V is the set of all vertices and E is the set of all edges, and given two disjoint subsets of vertices, namely, $M \subset V$, $N \subset V$ and $M \cap N = \emptyset$, where there exists at least k-vertex disjoint paths from each vertex $v \in N$ to the set of vertices M, find the set of edges F such that $G(V, E - F)$ satisfies the following:

1) There exist at least k-vertex disjoint paths from each vertex $v \in N$ to the set of vertices M.
2) $\sum_{i=1}^{N} p_i$ is minimized, where p_i is the weight of the maximum weighted edge $\in (E - F)$ of $v_i \in N$.

3.3 Disjoint Path Vector Algorithm for k-Vertex Supernode Connectivity

Our DPV algorithm efficiently assigns transmission power levels for sensor nodes while preserving the network's k-vertex supernode connectivity in a distributed and localized manner. It consists of five main stages where the first stage is path information collection, initiated by the supernodes through $Init$ messages. An $Init$ message contains the ID of the supernode that created the message and can only be transmitted by a supernode. These messages are received by the sensor nodes in the network and each receiver node updates its local path information according to that data. Sensor nodes transmit $PathInfo$ messages when an update occurs in their local disjoint path lists. Upon receiving a $PathInfo$ message, each sensor node computes the disjoint paths to the supernodes by using its local data and the path information received from the $PathInfo$ message. If the incoming $PathInfo$ message decreases the cost of the disjoint paths, the message is forwarded by adding the updated path information. The cost of a set of disjoint paths is defined as the maximum of the costs of the paths in the set. When further decrement is not possible, the first stage of the algorithm ends and the second stage starts in which each node calculates its required neighbors using the locally found set of disjoint paths as the input.

In the third stage, for each disjoint path a $Notify$ message is initiated by each sensor node and forwarded along the nodes in the path. Neighbor nodes in a disjoint path mark

TABLE 1
DPV Notations

T and T'	Set of local paths
D and D'	Set of disjoint paths with the minimum cost
U	Union of two sets of paths
I	Received PathInfo message
L	Maximum number of paths to be stored
c and c'	Cost of disjoint paths
r, p and p_i	Variables referencing paths
k	Degree of disjoint connectivity
R	Set of required neighbors
S	Set of first k disjoint paths
B	Set of neighbors of a node
Φ	Incoming Notify message
n	Variable referencing a node

each other as a required neighbor. In the fourth stage, neighbors that are not selected as required are removed from the neighbor list. In the final stage, each node adjusts its transmission power level to reach its farthest required neighbor according to the new topology that resulted from the removal of the non-required nodes. We now give a detailed explanation for each stage of the DPV algorithm.

After deploying sensor nodes and supernodes, the first stage of the algorithm is initiated by $Init$ messages that supernodes broadcast through the network. Any sensor node receiving an $Init$ message updates its local path information table by adding an entry for the path to the supernode that sent the message. In this special case such an entry will only include the ID of the supernode and the cost of the link between the receiver node and supernode. The cost will be the length of the link. An entry in a sensor node's local path information table at a sensor node includes a path to a supernode and the cost of that path. The cost of a path is the length of the longest link in the path. Paths are sorted according to cost in ascending order in the local path information table. Details of the distributed algorithm run by each sensor node in the information collection stage are given in Algorithm 3.1. The DPV algorithm notations are introduced in Table 1.

Algorithm 3.1 Path Information Collection in DPV

Input: I, L, k
Output: D
1: $T \leftarrow \emptyset$;
2: **for all** received PathInfo message I **do**
3: **if** I.Sender is a supernode **then**
4: $r \leftarrow$ new Path(I.Sender);
5: **if** $r \notin T$ **then**
6: $T \leftarrow T \cup r$;
7: Transmit PathInfo(T);
8: **end if**
9: **else**
10: $D \leftarrow$ MIN_DIS_SET(T);
11: $c \leftarrow$ Cost($\bar{D}$);
12: $U \leftarrow I.T \cup T$;
13: Sort(U);
14: $T' \leftarrow \{p_i \in U \mid i <= L\}$;
15: $D' \leftarrow$ MIN_DIS_SET(T');
16: $c' \leftarrow$ Cost($\bar{D'}$);
17: **if** $c' < c$ **then**
18: $T \leftarrow T'$;
19: Transmit PathInfo(T);
20: **end if**
21: **end if**
22: **end for**

Having updated the local path information table, a sensor node creates and transmits a $PathInfo$ message, which contains its ID and path information table. A $PathInfo$ message can only be sent by sensor nodes whereas an $Init$ message can only be sent by supernodes. $Init$ messages contain only ID of the sending supernode whereas a $PathInfo$ message includes paths to supernodes and their costs. Sensor nodes transmit $PathInfo$ messages to their reachable neighborhood using maximum power. Any sensor node that receives a $PathInfo$ message calculates the union of the existing and the received paths via $PathInfo$ message. Then, the set of disjoint paths with the minimum cost is calculated for the existing paths and for the newly calculated union. Notice that the size of calculated disjoint sets is at most k, because we need only k disjoint paths. If the new cost is lower than the current cost, the local path information table is updated. The resulting list of disjoint paths is stored for use in the next stages of the DPV algorithm. The details of the algorithm for finding disjoint paths (MIN_DIS_SET) run by each sensor node are given in Section 3 of the supplementary file, available online.

Any sensor node that updates its local path information table transmits a $PathInfo$ message containing the updated path information table. Note that if a $PathInfo$ message does not cause an update in the receiver node's disjoint paths list, then no $PathInfo$ message is sent by that sensor node. This condition ensures the termination of the path information collection stage. If further improvement on the path costs is not possible, then transmission of $PathInfo$ messages ends. The DPV algorithm is guaranteed to converge because there is an upper bound on the number of $PathInfo$ messages that any sensor can send during the path information collection stage. A $PathInfo$ message is only transmitted if an update occurs in the local set of disjoint paths. An update can only occur if a lower-cost disjoint path is discovered via a received $PathInfo$ message. In the worst case, there can be $|E|$ different-cost disjoint paths from a sensor node to the set of supernodes where $|E|$ is the number of edges in the network given that the cost of a path is defined as the cost of the longest edge in that path. In the worst case scenario, these $|E|$ different-cost disjoint paths can be discovered in the decreasing order of the path cost. Suppose that each discovery results an update, then the node will send at most $|E|$ $PathInfo$ messages. Since a graph with n nodes can have at most $O(n^2)$ edges, we conclude that any sensor node can transmit at most $O(n^2)$ $PathInfo$ messages. However, this upper bound occurs only when there is an edge between every node in the network, which is an uncommon scenario in WSNs. This upper bound can be better expressed as $O(n\Delta)$, where Δ denotes the maximum degree of a sensor node.

We set an upper limit on the number of hops a $PathInfo$ message can traverse during the path information collection stage. This is the $PathInfo$ message's time-to-live (TTL) value, and it directly affects the number of hops in the resulting paths. This value is set at the supernode that sends the $Init$ message and it is included in every $PathInfo$ message. When a sensor node receives a $PathInfo$ message that has reached the TTL value, no further $PathInfo$ message is created even if an update occurs as a result of this $PathInfo$ message. This approach ensures that paths with a length

above a certain limit are not considered in the disjoint path calculation; it can be considered as a prepruning that occurs before determining the set of disjoint paths with minimum cost. The Algorithm for finding the required nodes, which is run by each sensor node, is given in Algorithm 3.2.

Algorithm 3.2 Finding Required Neighbors

Input: D and k
Output: R
1: $R \leftarrow \emptyset$;
2: $S \leftarrow \emptyset$;
3: **if** $|D| >= k$ **then**
4: Sort(D);
5: $S \leftarrow \{p_i \in D \mid i <= k\}$;
6: **for all** $p \in S$ **do**
7: $R \leftarrow R \cup p$.First;
8: **end for**
9: **end if**
10: **for all** $p \in S$ **do**
11: Transmit Notify(p);
12: **end for**

To guarantee that all nodes in a selected disjoint path are labeled as required neighbors, we need to notify all the nodes on that path. To achieve this, each node sends a $Notify$ message for each of its selected disjoint paths. A $Notify$ message is forwarded along the disjoint path for which it was created. Each neighboring node in the disjoint path marks each other as required neighbors. This stage ensures that any node on a selected disjoint path will be marked as a required neighbor of its neighbors that are also on the same disjoint path. If any two neighbor nodes do not mark each other as required neighbors, it means that the link between these two nodes is not necessary and can be removed. Algorithm 3.3 shows the steps taken by each sensor node upon receiving a $Notify$ message.

Algorithm 3.3 Updating Required Neighbors By $Notify$ Messages

Input: R, B and Φ
Output: Updated R
1: **for all** received Notify message Φ **do**
2: **for all** Node $n \in \Phi$.Path **do**
3: **if** $n \in B$ **then**
4: $R \leftarrow R \cup n$;
5: **end if**
6: **end for**
7: **end for**

For a node that does not have path information of at least k-vertex disjoint paths to supernodes, all neighbors are defined to be required because it is not known which neighbors can be removed. Such a node cannot decide which neighbors are required and which are not, because it has not sufficient local information. In that case all neighbors are kept since all can potentially be a part of a disjoint path. This approach ensures the k-vertex supernode connectivity of the resulting network topology in case of insufficient local information. Having determined the final set of neighbors, each node adjusts its transmission power to reach its farthest neighbor according to the finished topology. A sample run of DPV is given in Section 1 of the supplementary file, available online. We prove that final topologies generated by the DPV algorithm are k-vertex supernode connected in Section 2 of the supplementary file, available online.

TABLE 2
Time and Message Complexities of DPV and DATC per Node Where n is the Number of Nodes in the Network, Δ is the Maximum Degree of a Sensor Node, and h is the Number of Hops that Specifies the Local Neighborhood of a Node in DATC

Algorithm	Time Complexity	Message Complexity
DPV	$O(n\Delta^2)$	$O(n\Delta)$
DATC	$O(\Delta^5)$	$O(\Delta^h)$

The running time complexity of the DPV algorithm is $O(n\Delta^2)$, where n is the number of nodes in the network and Δ is the maximum degree of a sensor node. In [35], the time complexity of the DATC algorithm is given as $O(\Delta^5)$. So, in dense graphs where Δ is high, DATC will experience much higher computation overhead than the DPV algorithm. In the worst case, where there is an edge between every node, complexity of DATC will be $O(n^5)$ whereas our algorithm will be $O(n^3)$. In our experiments we measured the running time of DPV as 4.5 seconds and the running time of DATC ($h = 2$) as 15.1 seconds on the average (we set the number of sensor nodes to 500). This result is in conformity with our running time analysis given in Section 4 of the supplementary file, available online.

The asymptotic message complexity of the DPV algorithm is $O(n\Delta)$. The message complexity of DATC is reported as $O(\Delta)$ per node in [35]. However this is valid for 1-hop neighborhood only. For h-hop neighborhood the complexity will be $O(\Delta^h)$, because each sensor needs to forward the messages that it receives from its h-hop neighborhood. On the other hand, one can note that the worst case for the DPV algorithm, where paths are discovered in the order of decreasing cost, is unlikely, since it is more probable to receive messages first from shorter paths than the longer ones. Therefore, we expect a lower message complexity in the average case. Our simulation results are in conformity with this analysis. Message complexity analysis of the DPV algorithm is given in Section 5 of the supplementary file, available online. Time and message complexity of the algorithms are summarized in Table 2.

4 Evaluation

In this section we report the results of the experiments that we performed to evaluate our algorithm, comparing it with GATC and DATC. We have implemented the DPV, GATC and DATC algorithms using a custom simulator that we developed. The simulator enables us to generate random network topologies, execute the algorithms on the generated topologies, calculate the desired metrics, and visualize the outputs of the experiments.

4.1 Experimental Setup

In our experiments, the sensors are randomly deployed in a 600 m $\times$ 600 m area. The supernodes are also randomly and uniformly deployed in this area. The path loss exponent for the wireless channel α is chosen to be 2 and the initial maximum transmission range R_{max} of the nodes is set to be 100 m. Since it is required to have a network topology where each sensor is at least k-vertex connected to the set of supernodes, topologies that do not satisfy this condition are discarded. For the parameters k and h we use similar values

TABLE 3
Simulation Parameters

Deployment Area	$600m \times 600m$
Path Loss Exponent: α	2
Initial Transmission Range of Nodes: R_{max}	$100m$
Number of Sensors: N	[100...500]
Number of Supernodes: M	5% and 10% of N
Degree of Disjoint Connectivity: k	2 and 3
Number of Hops for Neighborhood in DATC: h	1 and 2
Confidence Level	95%

with [35], which are typical values seen in k-connectivity studies. The experiments are repeated at least 50 times for each parameter set and average values are reported at 95% confidence level with a maximum 5% confidence interval. Confidence intervals for reported values are depicted on the corresponding charts. Our simulation parameters are summarized in Table 3.

We assume that each node in the network has a unique ID and for the first part of the simulations we assume no collisions or retransmissions occur during wireless communications. Hence we only focus on topology and distance effects. Then we extend our simulations with packet loss scenarios in Section 4.2.5. In addition, we assume that nodes can predict the length of the links using received signal strength. As sensor nodes we consider TelosB motes which have an adjustable transmission range where R_{max} is 100 m. As supernodes we consider Crossbow IRIS motes which can have transmission ranges up to 300 m. Both nodes are Zigbee/802.15.4 compliant so sensor nodes can both communicate with each other and also with supernodes. The maximum transmission range for supernodes to communicate with sensor nodes is set to be 100 m as well. For sensor nodes, the transmit power needed to talk to a receiver in range is calculated by assuming a simple path loss model and using the simple formula $p_i = r_i^{\alpha}$, as done in [35]. Here, p_i is the transmit power node i is using, r_i is the transmission range of the node, and α is the path loss exponent. Hence, the power values are simply expressed as a function of the transmission range (i.e., distance) for the purpose of comparing the algorithms. Therefore, we do not give power values in a specific energy unit such as Joule.

4.2 Results

4.2.1 Total Transmission Power

In this section we report our simulation experiment results showing the total transmission power set for the sensor nodes in the resulting topologies obtained after executing GATC, DATC with 1-hop local neighborhood ($h = 1$), DATC with 2-hop local neighborhood ($h = 2$) and DPV. GATC's results are presented as a reference point, since it minimizes the maximum transmission power of sensor nodes. GATC can achieve that since it has the global network topology information by being a centralized algorithm.

In Figs. 1a and 1b, we present the total transmission power in topologies with $k = 2$. In Fig. 1a, the number of sensor nodes increases from 100 to 500 and the number of supernodes is set to be 5% of the number of sensor nodes. We see that the total transmission power in the topologies generated by our DPV algorithm is much lower than that of DATC ($h = 1$) and DATC ($h = 2$) algorithms. This shows that the link elimination approach utilized by DATC has difficulties in discovering alternative disjoint paths to a long direct edge using local data gathered from 1 or 2-hop neighborhood, as we discussed in Section 2. Since the search scope is limited, long, hence, costly edges cannot be substituted with cheaper disjoint paths, resulting long transmission ranges and thus high total transmission power. Another important observation related to limited search scope is that DATC ($h = 1$) performs significantly worse than DATC ($h = 2$) which has a broader search scope. However, as will be seen in later figures, DATC ($h = 2$) requires significantly more control message transmissions compared to DATC ($h = 1$) for maintaining 2-hop neighborhood.

Fig. 1b shows the results of our experiments where the number of supernodes is 10% of number of sensor nodes. In Fig. 1b, we see almost the same picture as in Fig. 1a, except that the total power values are little lower because, due to the high number of supernodes in the network, it is more likely that cheaper paths will be found. Our DPV algorithm performs parallel to GATC and the difference between DPV and DATC ($h = 2$) becomes clearer as the network gets denser.

Figs. 1c and 1d depict the total transmission power of the topologies generated by the algorithms for $k = 3$. We see that the total transmission power of the topologies generated by DPV is significantly lower than that of DATC ($h = 1$) and DATC ($h = 2$). For $k = 3$, it is harder to substitute a high cost edge with a low cost one because three disjoint paths are needed to be able to do that. DATC needs to find these paths in its reachable neighborhood where our algorithm can directly search for paths using information in the path vectors. Therefore, our algorithm can achieve lower total transmission power compared to DATC. We also observe that satisfying 3-vertex disjoint supernode connectivity requires more transmission power than satisfying 2-vertex disjoint supernode connectivity in general, as expected.

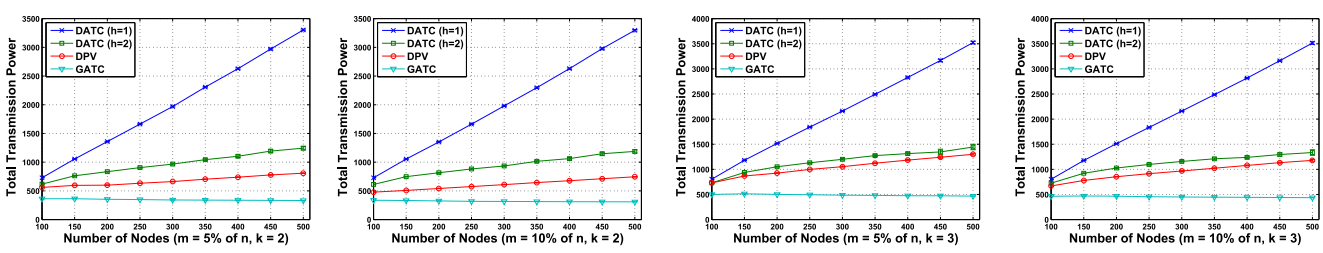

Fig. 1. Total transmission power.

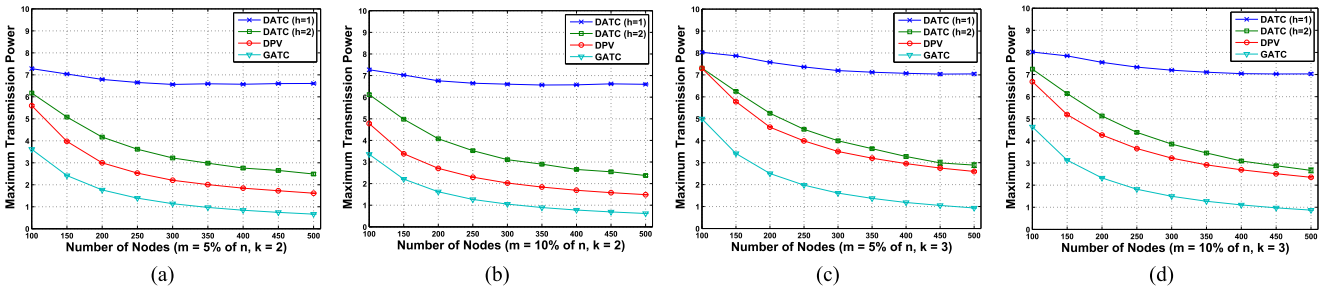

Fig. 2. Maximum transmission power.

4.2.2 Maximum Transmission Power

In this section, we report our experiment results on maximum transmission power among all sensors in the resulting topologies generated by the DATC, DPV and GATC algorithms. Maximum transmission power is an important metric for the resulting topologies because it is an indication of the balance of energy consumption among all sensors. Even if total transmission power is low, the resulting topology may become disconnected or even partitioned if the maximum transmission power is high because sensor nodes with a high transmission range will use more energy than other nodes and deplete their batteries sooner.

Figs. 2a and 2b show the maximum transmission powers of the topologies generated by the algorithms for $k = 2$, where the number of supernodes is 5 and 10% of the sensor nodes, respectively. We observe that GATC produces topologies with the lowest maximum transmission power among all algorithms. This is due to the fact that GATC is a centralized algorithm and has global network topology information. Thus it directly knows where the longest edges reside in the network and eliminates edges in the order of decreasing cost when possible. The DPV and DATC algorithms, on the other hand, are localized algorithms and have a partial knowledge of the network topology. For that reason, they generate topologies with higher maximum transmission power compared to GATC. Our proposed DPV algorithm has the next best performance, which is significantly better than the DATC algorithms. This result is again related to the available information for discovering disjoint paths. DATC can obtain limited information compared to DPV, therefore it cannot discover as many disjoint path as our DPV algorithm. This results higher maximum transmission ranges for DATC.

For $k = 3$, Figs. 2c and 2d show the comparison of maximum transmission powers for the resulting topologies. The results are similar to when $k = 2$ except that maximum transmission power values are greater. This is an expected result because keeping the network 3-connected requires keeping more neighbors connected, which leads to higher transmission ranges for the sensor nodes.

4.2.3 Total Number of Control Message Transmissions

In this section we give and discuss our experiment results regarding the total number of control message transmissions during the execution of the DPV and DATC algorithms. GATC does not require control message transmissions while computing the topology, since the computation is done centrally. Control message transmission metric is important because it is necessary not only to consider power consumption in the resulting topologies but also to consider the power consumption required to generate those topologies, which can be viewed as a fixed cost of obtaining the final topologies. If this cost is high, then the power efficiency of the resulting topology might become meaningless.

In Figs. 3a and 3b, we present the total number of required control message transmissions to execute the algorithms with $k = 2$. The most important observation here is that DATC ($h = 2$) requires an asymptotically larger number of transmissions than DATC ($h = 1$) or DPV, because in DATC ($h = 2$) each node needs to notify all nodes in its 2-hop neighborhood. While DATC ($h = 1$) and DPV require only one message transmission for an update, DATC ($h = 2$) requires Δ transmissions, where Δ is the degree of a given sensor node. This situation shows that DATC ($h = 2$) may not be feasible in practice, because transmitting so many messages during algorithm execution will cause rapid battery drain. Therefore, extending the DATC algorithm for more than 1-hop neighborhood seems not to be scalable due to the exponentially increasing number of update messages.

We also observe that DPV requires fewer transmissions than DATC ($h = 1$) to obtain a k-vertex supernode connected

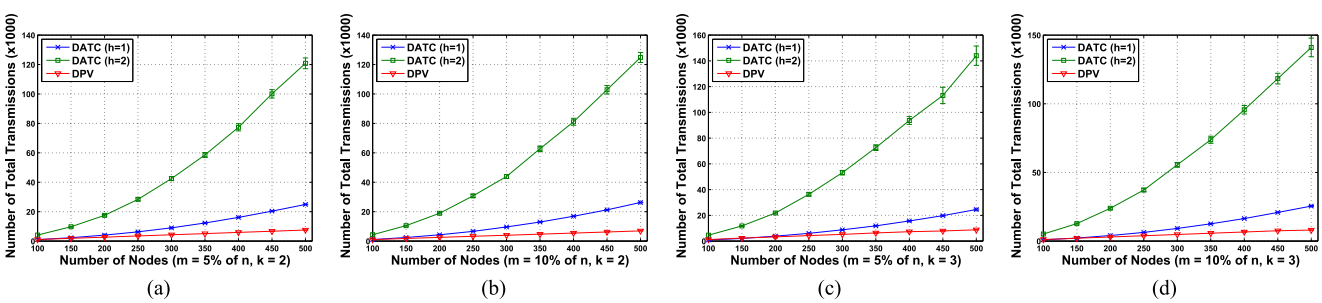

Fig. 3. Total number of transmissions.

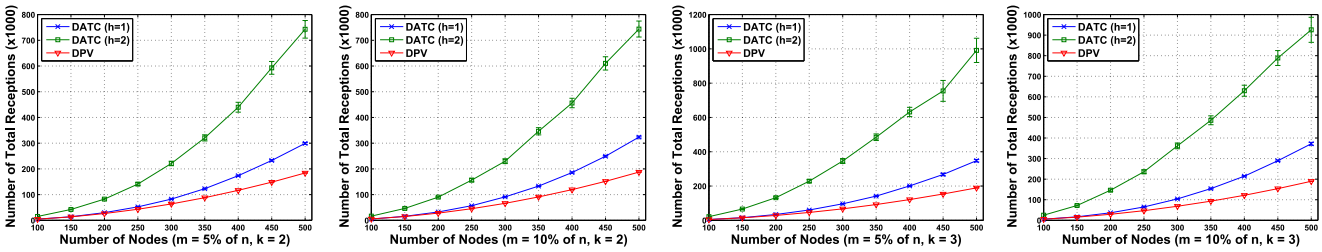

Fig. 4. Total number of receptions.

network topology. This is expected because DPV transmits an update message only if a set of disjoint paths with a lower cost is discovered whereas DATC sends update messages in all power increments. Due to smaller search scope, DATC is often unable to find disjoint paths and thus the number of power increments tends to be high.

In Figs. 3c and 3d, we present the total number of required control message transmissions to execute the algorithms with $k = 3$. The behaviors of all three algorithms are similar with $k = 2$, except when $k = 3$ all algorithms require slightly more message transmissions.

4.2.4 Total Number of Control Message Receptions

Another factor that affects power efficiency is the total number of received control messages during the execution of the topology control algorithms. This is again a fixed cost, as in the case of total transmissions required.

Figs. 4a and 4b show the total number of received control messages during the execution of the algorithms with $k = 2$, and Figs. 4c and 4d show the same for $k = 3$. Here again we observe that during the execution of DATC ($h = 2$) many more receptions occur compared to DATC ($h = 1$) and DPV in all cases. This is a direct result of high number of transmissions required by DATC as discussed in Section 4.2.3.

We observe that when the number of supernodes is 5% of the sensor nodes and $k = 2$, DPV requires fewer receptions than DATC ($h = 1$). In all other cases the total number of received messages for DPV and DATC ($h = 1$) is almost the same. For all algorithms, number of receptions tend to increase when k and the number of supernodes increase, but the increase for DATC ($h = 2$) is significantly faster than that of DPV and DATC ($h = 1$). Another important observation is that the number of receptions is about five times greater than the number of transmissions. This result is expected because when a message is transmitted it is received by multiple nodes in the 1-hop neighborhood.

4.2.5 Effect of Packet Losses

In wireless transmission medium, packet losses may be frequent due to collisions, link failures, high bit error-rates, and environmental conditions. In order to justify our results, we present an additional scenario where packet losses can occur in any link in the network with a given probability. In this scenario, retransmissions are also considered and maximum number of retransmissions is set to be 3. We change the packet loss ratio (PLR) between 0.1 and 0.4 with a step size of 0.1, and present the effect of packet losses on total transmission power in Figs. 5a, 5b, 5c, and 5d.

We observe that when PLR increases, total transmission power also increases. This is an expected result because under a higher PLR, a higher number of information exchange messages are lost and thus nodes can obtain incomplete topology information from their neighbors. Using this partial information all three algorithms miss some of the more efficient paths because messages carrying that information are lost due to communication failures.

As seen in Figs. 5a and 5b, our DPV algorithm performs better than both DATC ($h = 1$) and DATC ($h = 2$), as in the previous scenarios that do not consider packet losses. However, we see that the reduction in total power is not as high as in the previous scenarios; it drops to 30% when PLR is 0.2. When PLR is 0.3 or higher, DATC ($h = 2$) starts performing better than our algorithm in terms of total transmission power of the final topologies. This is due to the fact that DATC algorithms are more localized than DPV because they only use 1-hop or 2-hop neighborhood information which means their messages go only 1- or 2-hops long. On the other hand, DPV algorithm gathers path information from a wider area and therefore control messages can traverse up to a predefined TTL value which is typically six or seven. Since messages traverse longer paths in DPV algorithm, they are more likely to get lost. To elaborate, successful delivery chance of a message over a 2-hop path is three

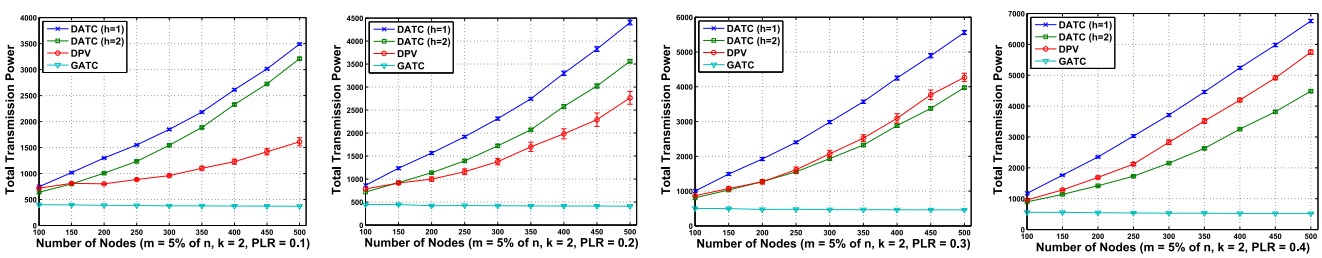

Fig. 5. Total transmission power for $k = 2$ with packet loss.

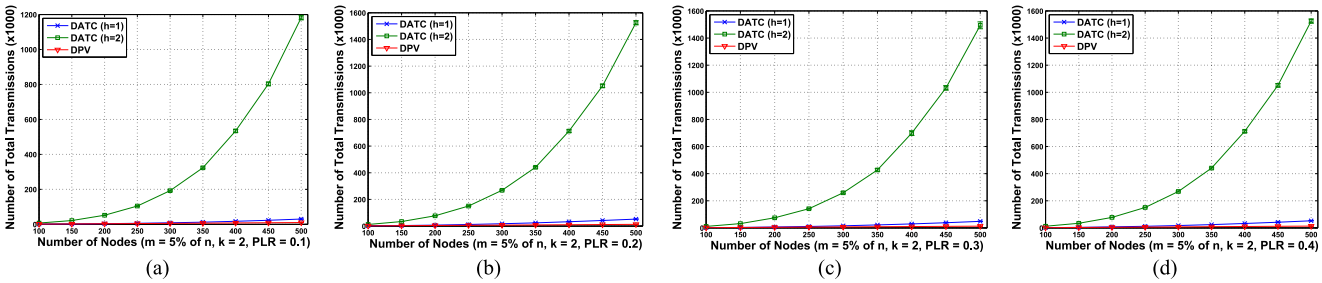

Fig. 6. Total number of transmissions for $k = 2$ with packet loss.

times higher compared to a 5-hop path when PLR is 0.3. However, a PLR value as high as 0.3 per link is not very realistic. In [33], it is reported that the average PLR is below 0.1 for Crossbow IRIS motes when the transmitter-receiver separation is between 0 and 70 meters. In addition, total number of control message transmissions for DATC ($h = 2$) is much higher than DPV due to the need of power update messages in 2-hop neighborhood, as seen in Figs. 6a, 6b, 6c, and 6d. DATC ($h = 1$) still results with the highest total transmission power due to its limited search scope as discussed in Section 4.2.1. Therefore, we can conclude that under realistic PLR values, our algorithm DPV still outperforms DATC algorithms in terms of total transmission power and total number of transmissions.

5 Conclusion

In this paper we introduce a new distributed and fault-tolerant algorithm, called Disjoint Path Vector Algorithm (DPV), for constructing fault-tolerant topologies for heterogeneous wireless sensor networks consisting of supernodes and ordinary sensor nodes. Our algorithm results in topologies where each sensor node in the network has at least k-vertex disjoint paths to the supernodes. The objective of the algorithm is to minimize the total transmission power of the nodes in the network.

Through extensive simulations, we show that our approach outperforms the existing algorithms in terms of energy efficiency. Compared to an existing solution, the DATC algorithm, our DPV algorithm achieves a 4-fold reduction in total transmission power and a 2-fold reduction in maximum transmission power under the assumption of no packet losses. When we consider the packet losses, 2.5-fold reduction in total power consumption can be achieved for a packet loss rate of 0.1. In addition, DPV achieves these results by requiring fewer message transmissions and receptions than DATC.

The most important contribution of this study is to generate topologies that have total transmission powers close to GATC, which is a centralized algorithm. As mentioned before, GATC requires global network information and hence, it is less practical for large-scale networks. The solution that we propose is, however, distributed and localized, thus is scalable to large networks and therefore suitable for use in real applications.

References

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, 2008.

[2] G. Anastasi, M. Conti, M. Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 537–568, 2009.

[3] H. Liu, A. Nayak, and I. Stojmenovic, "Fault-tolerant algorithms/protocols in wireless sensor networks," *Guide Wireless Sensor Netw.*, pp. 261–291, Chapter 10, 2009, DOI: 10.1007/978-1-84882-218-4_10

[4] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Comput. Surveys*, vol. 37, no. 2, pp. 164–194, 2005.

[5] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Netw.*, vol. 2, no. 4, pp. 351–367, 2004.

[6] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks, in," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2005, vol. 2, pp. 878–890.

[7] Y. Wang, "Topology control for wireless sensor networks," in *Wireless Sensor Netw. Appl.*, pp. 113-147, Chapter 5, 2008, DOI: 10.1007/978-0-387-49592-7_5

[8] X. Wang, M. Sheng, M. Liu, D. Zhai, and Y. Zhang, "RESP: A k-connected residual energy-aware topology control algorithm for ad hoc networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2013, pp. 1009–1014.

[9] L. Li, J. Halpern, P. Bahl, Y. Wang, and R. Wattenhofer, "A cone-based distributed topology control algorithm for wireless multi-hop networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 147–159, Feb. 2005.

[10] N. Li and J. C. Hou, "FLSS: A fault-tolerant topology control algorithm for wireless networks," in *Proc.10th ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2004, pp. 275–286.

[11] N. Li and J. C. Hou, "Localized fault-tolerant topology control in wireless ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 4, pp. 307–320, Apr. 2006.

[12] L. Wang, H. Jin, J. Dang, and Y. Jin, "A fault tolerant topology control algorithm for large-scale sensor networks," in *Proc. 8th Int. Conf. Parallel Distrib. Comput, Appl. Technol*, 2007, pp. 407–412.

[13] D. M. Blough, M. Leoncini, G. Resta, and P. Santi, "The k-neigh protocol for symmetric topology control in ad hoc networks," in *Proc. 4th ACM Int. Symp. Mobile ad hoc Netw. Comput.*, 2003, p. 152.

[14] Y. Chen and S. H. Son, "A fault-tolerant topology control in wireless sensor networks," in *Proc. 3rd ACS/IEEE Int. Conf. Comput. Syst. Appl.*, 2005. p. 57.

[15] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Netw.*, vol. 8, no. 5, pp. 481–494, 2002.

[16] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2004, pp. 144–158.

[17] C. Hua and T.-S. P. Yum, "Asynchronous random sleeping for sensor networks," *ACM Trans. Sensor Netw.*, vol. 3, no. 3, pp. 1–15, 2007.

[18] Y. M. Baryshnikov, E. G. Coffman, and K. J. Kwak, "High performance sleep-wake sensor systems based on cyclic cellular automata," in *Proc. Int. Conf. Inform. Process. Sensor Netw.*, 2008, pp. 517–526.

[19] O. Younis and S. Fahmy, "HEED: A hybrid, energy efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 366–379, Oct. 2004.

[20] N. Li, J. C. Hou, and L. Sha, "Design and analysis of an MST-based topology control algorithm," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, pp. 1195–1206, May 2005.

[21] X. Li, Y. Wang, and W. Song, "Applications of k-local MST for topology control and broadcasting in wireless ad hoc networks," *IEEE Trans. Parallel Distrib Syst*, vol. 15, no. 12, pp. 1057–1069, Dec. 2004.
[22] W. Wu, H. Du, X. Jia, Y. Li, and S. C.-H. Huang, "Minimum connected dominating sets and maximal independent sets in unit disk graphs," *Theoretical Comput. Sci.*, vol. 352, no. 1, pp. 1–7, 2006.
[23] J. Wu, F. Dai, M. Gao, and I. Stojmenovic, "On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks," *IEEE/KICS J. Commun. Netw.*, vol. 4, no. 1, pp. 59–70, Mar. 2002.
[24] W.-Z. Song, Y. Wang, X.-Y. Li, and O. Frieder, "Localized algorithms for energy efficient topology in wireless ad hoc networks," in *Proc. ACM Int. Symp. Mobile Ad-Hoc Netw. Comput.*, 2004, pp. 98–108.
[25] W.-Z. Song, X.-Y. Li, O. Frieder, and W. Wang, "Localized topology control for unicast and broadcast in wireless ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 4, pp. 321–334, Apr. 2006.
[26] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: Recent developments and deployment challenges," *IEEE Netw.*, vol. 20, no. 3, pp. 20–25, May/Jun. 2006.
[27] K. K. Mamidisetty, M. J. Ferrara, and S. Sastry, "Systematic selection of cluster heads for data collection," *J. Netw. Comput. Appl.*, vol. 35, pp. 1548–1558, 2012.
[28] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proc. 22nd Annu. Joint Conf. IEEE Comput. Commun.*, 2003, vol. 3, pp. 1713–1723.
[29] M. Azharuddin, P. Kuila, and P. K. Jana, "A distributed fault-tolerant clustering algorithm for wireless sensor networks," in *Proc. IEEE Int. Conf. Adv. Comput., Commun. Inform.*, 2013, pp. 997–1002.
[30] J. Wu, S. Yang, and M. Cardei, "On maintaining sensor-actor connectivity in wireless sensor and actor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2008, pp. 888–896.
[31] K. Ozaki, K. Watanabe, S. Itaya, N. Hayashibara, T. Enokido, and M. Takizawa, "A fault-tolerant model for wireless sensor-actor system," in *Proc. 20th Int. Conf. Adv. Inform. Netw. Appl.*, vol. 2, Apr. 2006, p. 5.
[32] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz, "A distributed coordination framework for wireless sensor and actor networks," in *Proc. 6th ACM Int. Symp. Mobile ad hoc Netw. Comput.*, 2005, pp. 99–110.
[33] C. Cirstea, M. Cernaianu, and A. Gontean, "Packet loss analysis in wireless sensor networks routing protocols," in *Proc. 5th Int. Conf. Telecommun. Signal Process.*, Jul. 2012, pp. 37–41.
[34] Z. Gengzhong and L. Qiumei, "A survey on topology control in wireless sensor networks," in *Proc. 2nd Int. IEEE Conf. Future Netw.*, 2010, pp. 376–380.
[35] M. Cardei, S. Yang, and J. Wu, "Algorithms for fault-tolerant topology in heterogeneous wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 4, pp. 545–558, Apr. 2008.
[36] M. Haghpanahi, M. Kalantari, and M. Shayman, "Topology control in large-scale wireless sensor networks: Between information source and sink," *Ad Hoc Netw.*, vol. 11, no. 3, pp. 975–990, 2012.
[37] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel, "Topology management techniques for tolerating node failures in wireless sensor networks: A survey," in *Comput. Netw.*, vol. 58, pp. 254–283, 2014.

Hakki Bagci received the BS and MS degrees in computer science from Bilkent University, Ankara, Turkey, in 2004 and 2007, respectively. He received the PhD degree from the Department of Computer Engineering, Middle East Technical University. He has worked as a researcher at the National Scientific and Technological Research Council of Turkey (TUBITAK) since 2004. His research interests include distributed algorithm design for wireless sensor networks.

Ibrahim Korpeoglu received the BS degree (summa cum laude) in computer engineering, from Bilkent University in 1994, the MS, and PhD degrees from the University of Maryland at College Park, both in computer science, in 2000 and 1996, respectively. He is an associate professor in the Department of Computer Engineering, Bilkent University, Ankara, Turkey. Since 2002, he is a faculty member in the Department of Computer Engineering of Bilkent University. Before that, he worked in several research and development companies in USA including Ericsson, IBM T.J. Watson Research Center, Bell Laboratories, and Bell Communications Research (Bellcore). He received Bilkent University Distinguished Teaching Award in 2006 and IBM Faculty Award in 2009. He is a member of the ACM and a senior member of the IEEE.

Adnan Yazıcı received the PhD degree in computer science from the Department of EECS, Tulane University, in 1991. He is a full professor and chairman in the Department of Computer Engineering, METU, Ankara, Turkey. His current research interests include intelligent database systems, spatio-temporal databases, multimedia and video databases, and wireless multimedia sensor networks. He has published more than 180 international technical papers. He has received IBM Faculty Award for 2011 and Young Investigator Award bestowed by the Parlar Foundation, for the year 2001. He was a conference co-chair of the 23rd IEEE International Conferences on Data Engineering in 2007, the 38th Very Large Data Bases in 2012 and a program co-chair of Flexible Query Answering Systems in 2011. He is the director of Multimedia Database Laboratory at METU. He is a member of the ACM and a senior member of the IEEE.