Abstract—Topology control is important for heterogeneous sensor networks (HSNs) in order to minimize the total network power consumption under the constraint that all sensor nodes’ connectivity requirements are satisfied. To address this issue, an optimization problem is first formulated, which is formally proved to be NP-hard. For practical applications, an effective solution, named topology adaptation algorithm (TAA), is proposed. TAA adopts both graph theory and maximum flow theory to find pre-specified node disjoint paths with low time complexity and high network power efficiency. In order to further save the network power consumption, a judgment theory is proposed to remove any unnecessary long edges at the beginning without affecting network connectivity. Both theoretical and numeric results show that the proposed topology control algorithm can outperform counterparts in terms of the total network power consumption, the percentage of supernodes achieving \(k\)-connectivity, the average degree of nodes, and the average length of paths.

Index Terms—sensor network, topology control, heterogeneous connectivity, maximum flow, judgment theory.

I. INTRODUCTION

Wireless sensor networks (WSNs) have demonstrated their huge application potentials in practice, such as military surveillance, environmental monitoring, target tracking, disaster rescue, and smart homes. Commonly, WSNs consist of various resource-constrained (in terms of energy, communication and computation resources) sensor nodes.

These nodes, without human intervention, are capable of collaborating with each other to collect, process and deliver environmental information, make decisions and perform appropriate actions to accomplish complicated tasks.

Different from conventional wireless ad hoc networks (WANET), many realistic WSNs feature various sensors nodes [1], [2], called HSNs, which implies the request for vastly diverse services. For example, due to heterogeneous criticality in sensed data, different device nodes may request different connectivity services, or transmission reliability, to the sink. A practical application of HSNs is a home sensor network. In such a network, biomedical sensors that monitor patients’ physiological parameters collect vital medical signs, so that their data have to be transmitted to the sink timely and reliably even if there are failures on some network nodes. On the contrary, the data generated by smart meters have no such stringent requirements and can tolerate some transmission delay or packet loss during network failures.

In this paper, transmission reliability is defined as the network connectivity to the sink, i.e., a node being \(k\)-connected means that there are \(k\) node-disjoint paths from the node to the sink. With different values of \(k\), different levels of transmission reliability can be achieved. Besides, in sensor networks, two-way data transmission between any device node and the sink is necessary. The downstream links from the sink to device nodes may be used to send instructions to end-users, while the up-stream links from device nodes to the sink can send device states and sensed data. All of these raise the request on designing suitable topology control mechanisms for HSNs.

In the literature, topology control has been widely discussed for Internet [3]–[7] and ad hoc networks [8]–[23], [40], [41]. However, current research efforts on topology control in HSNs mainly concentrated on the heterogeneity of sensor nodes in
terms of power reserves, communication ranges, and sensing ranges [8]–[13]. In addition, most works for $k$-fault tolerant topology control algorithms were built on homogeneous connectivity requirement assumptions [3]–[7], [12], [14]–[16] and cannot be applied directly to HSNs. Moreover, the existing topology control mechanisms in Internet missed the consideration of issues on network power consumption [3]–[7], while for ad hoc networks, most of the research only considered the scenario that any pair of nodes are $k$-connected, and few efforts were dedicated to the all-to-one topology control problem.

Different from existing works, this paper focuses on topology control in HSNs in order to provide heterogeneous two-way connectivity between all device nodes and the sink and at the same time minimize network power consumption. For the purpose of explanation, we consider a HSN consisting of two types of nodes, called supernodes and ordinary nodes. Supernodes are required to be $k$-connected with the sink and ordinary nodes to be 1-connected. However, our work can be easily extended to the scenario with more than two types of nodes. Apparently, the $k$ node-fault tolerant topology control problem in conventional ad hoc networks is only a special case of our defined model, where all device nodes in the network are supernodes.

In this paper, we first formulate the topology control problem as an optimization problem, which minimizes the total power consumption under the constraint that all nodes’ connectivities are satisfied. Since the formulated optimization problem is formally proved to be NP-hard, we develop an approach, called TAA, to find a solution with reasonable computational complexity. Specifically, we introduce the concept of graph transformation to decouple the solution into two steps. In the first step, the network topology is transformed to a minimum-weight spanning tree so that the transformed graph is connected and all ordinary nodes meet their connectivity requirements. After that, in step two, a connectivity augmentation algorithm (CAA) is proposed to add more links into the previously transformed graph so that all supernodes have at least $k$ node-disjoint paths. The proposed algorithm transforms the original neighbor graph to a directed graph so that finding node-disjoint paths in the original graph becomes searching edge-disjoint paths in the latter by solving a maximum flow problem. After that, a greedy algorithm is proposed to select $k$ best paths, which introduces minimum extra power consumption. Through simulations, we notice that many long edges without affecting the network connectivity of any node are selected as candidate edges at the initial stage of TAA. To avoid this situation and further improve the power efficiency, we introduce and prove a judgment theorem, and based on this theory, we propose the preprocessing algorithm (PA) to remove unnecessary long edges in advance. Simulation results show that the proposed algorithm outperforms the counterparts in terms of the total network power consumption, the percentage of supernodes achieving $k$-connectivity, the average degree of nodes and the average length of paths. The main contributions of this paper are summarized as follows.

- We formulate an optimization problem for the topology control problem in HSNs, where multiple transmission reliability services are required. After formally proving the formulated optimization problem to be NP-hard, we propose an efficient topology control algorithm based on graph transformation to find pre-specified node disjoint paths for devices nodes from themselves to the sink with low time complexity and at the same time try to minimize power consumption.
- We propose a judgment theorem, which allows to remove some redundant edges without impacting the connectivity of nodes. Based on this theorem, we propose a PA to avoid unnecessary long edges being selected as candidate edges so as to further reduce the network power consumption.
- Theoretical analysis and simulation results demonstrate the combination of TAA and PA, called TAA+PA, can guarantee a feasible solution to the topology control problem with heterogeneous connectivity requirements. Simulation results also show the effectiveness of TAA+PA in terms of the total power consumption and the average degree of nodes.

The rest of this paper is organized as follows. In section II, we summarize the related works in literature. In section III, we present the system model under consideration, and formally prove the optimal topology control problem to be NP-hard. Section IV presents the proposed TAA in details.
Section V describes the proposed PA. In section VI and VII, we evaluate the proposed algorithm through theoretical analysis and simulation. Section VIII gives concluding remarks.

II. RELATED WORKS

In the literature, topology control in wireless sensor networks has been widely studied. Most works were under the assumption that all nodes and links are identical. Recent researches showed that allowing heterogeneity of sensor nodes can significantly improve performance for sensor networks [8]–[11], [13]. In [8], Li et al considered a HSN consisting of two different types of nodes, and proposed a clustering topology control model with local-world properties. Targeting at HSNs with unidentical communication ranges, authors in [9] introduced a rigid graph concept, and proposed an improved geographical adaptive fidelity algorithm to decrease energy consumption while guaranteeing the formed topology is 2-connected. Authors in [10] considered a scenario where a large portion of normal nodes had short communication radius while a small portion of super nodes had long communication ability, and presented a topology evolving model with small-world and scale-free features. Similar to [10], Guidoni et al [11] introduced the Kleinberg model for HSNs, and applied it to create shortcuts in the corresponding network graph to form a topology with small-world and scale-free features. Similar to [10], Guidoni et al[11] introduced the Kleinberg model for HSNs, and applied it to create shortcuts in the corresponding network graph to form a topology with small-world and scale-free features. In order to address a multi-objective optimization problem which incorporated topology control and coverage issues, Jameii et al [13] proposed an optimization approach based on genetic algorithm and learning automata. Notice that the definition of HSN in this paper is totally distinct from that in [8]–[11], [13].

In this paper, we focus on the heterogeneity of data criticality resulted from heterogeneous sensing sources.

There are research works on establishing fault-tolerant transmission paths in distributed networks [3]–[7], [31], [35]. Multipath routing and path splicing are two most commonly used techniques. Multipath routing aims at providing multiple disjoint paths between nodes. For example, the approach in [3] calculated the connectivity of two given nodes through repeatedly searching the shortest path. In [4], Nagamochi and Ibaraki applied maximum adjacency (MA) ordering to graph connectivity. An algorithm family, called maximum alternative routing algorithms (MARAs), was proposed in [5] to construct a directed acyclic graph with the objective of maximizing the minimum connectivity. Instead of finding the maximum number of routing paths, some studies targeted at establishing optimal multiple paths according to different routing metrics. Stefanescu et al [31] adopted an improved ad-hoc network on-demand distance vector protocol (AODV) to provide preferred multiple paths. In [32], the main routing metrics included link and node stability. However, for all these multipath routing schemes, i.e., the nodes may become unconnected if at least one link fail of each path, even if the underlying network remains connected. In order to achieve multipath routing against single-link failures, the authors in [6] introduced the concept of independent directed acyclic graphs (IDAGs), and utilized path augmentation techniques to achieve multipath routing. In [7], the authors explored the potential of path splicing to increase the fault tolerance, and proposed an approach to construct two routing trees against all single-link failures by using ear decomposition techniques. However, such approaches are not suitable for HSNs, since the corresponding node power consumption of the resultant network graph is not considered.

For ad hoc networks, existing topology control approaches mostly focused on reducing power consumption under the constraint of a given network connectivity [14]–[18], [20]–[23], [34]. In [14], Konstantinidis et al proposed a multiple-objective evolutionary algorithm to address the unified k-connected deployment and power assignment problem. In the formed topology, each node either can communicate with the sink, or has k neighbors towards the sink, and each grid point within the deployment zone is covered. In [15], Liu et al proposed a k-connectivity power assignment algorithm. The main idea of the proposed algorithm was to construct a full-load-operation network graph and delete long links that won’t affect node connectivity. By considering four topology control scenarios with respect to input graph (symmetric or asymmetric) and solution (unidirectional or bidirectional), R-Moraes et al [16] presented three mixed integer programming formulations for the k-connectivity power assignment problem. Guo et al [18] exploited the particle swarm optimization strategy to present k-connected fault-tolerant topology control. Saha et
al [22] proposed a distributed $k$-connected topology control algorithm, where any node maintains $k$ optimal node disjoint paths with each of its neighbor nodes to guarantee $k$-connectivity among them. In these graphs, any pair of vertices has $k$ node-disjoint paths. In [34], Bagci et al discussed a special type of sensor networks with two-layered architectures, and designed a $k$-connectivity topology control algorithm by storing full path information in local information tables, where node connectivity was defined as the number of node disjoint paths from sensor nodes to super nodes.

Some research works have been done on establishing 2-connected topology, where any pair of nodes are required to have dual node/arc disjoint paths. Carvalho et al [37] formulated the 2-connected topology control problem as an integer programming problem and a cut covering optimization problem, which were solved by using a commercial solver. RMoraes et al defined the minimum-power biconnected topology control problem as a mixed integer programming problem. Since current commercial solvers can only calculate the optima of the small-scale networks, they introduced a heuristic algorithm for the large-scale problem [17]. In [21], by utilizing connected dominating set, Wang et al proposed an augmentation algorithm to form a connected graph without cut-vertices, which means that the formed graph is 2-connected.

Targeted at all-to-one and one-to-all topology control problems in sensor networks, all nodes except the sink, are required to have $k$ node-disjoint paths from themselves to the sink. In [41], Li et al proposed a distributed tree-shaped $k$-connectivity topology control algorithm, where all first-level subtrees of the formed topology traversed the deployment area. However, the algorithm was built on some strong assumptions, e.g., the deployment area should be strip-shaped and the value of $k$ had to be rather small, and it can not guarantee finding a feasible solution even if it exists. Wang et al [21] exploited Ford-Fulkerson algorithm [33] to propose a minimum power based and an augmentation based approaches. The time complexity of both approaches is $O(n^9 \log n)$. Li et al [23] transformed the $k$-connected topology control problem into an integer flow problem, and analyzed the time complexity of the proposed algorithm, which is $O(n^8)$. Obviously, the time complexity of the aforementioned algorithms is too high for a practical implementation.

In order to decrease the time complexity, Wang et al [21] proposed an efficient heuristic algorithm (HA) to construct $k$ unconnected minimum weight subtrees by repeatedly constructing the minimum weight sink tree $k$ times, wherein the union of such $k$ subtrees implied $k$ node-disjoint paths from each sensor node to the sink.

### III. System Model and Problem Formulation

Consider a HSN consisting of one sink $t$ and many energy-constraint device nodes. The sink and all device nodes are randomly distributed over a target area. Since the data generated from each device node may have different criticality, devices nodes are assigned different reliability requirements in terms of connectivity. For explanation purpose, in this paper, all device nodes are categorized into two types, i.e., supernodes and ordinary nodes. Supernodes are required to have $k$ ($k > 1$) node-disjoint paths to the sink, while ordinary nodes need at least one path. We assume the network under consideration is feasible, i.e., at least $k$ node-disjoint paths exist for all supernodes. We define that there are $M$ supernodes and $N$ ordinary nodes. Note that our work can be easily extended to the situation with more than two connectivity requirements.

From topology control point of view, the defined HSN can be represented as a undirected flow network $G = (V, E)$, where the set of vertices, $V = \{v_i\}_{i=0}^{N+M}$, consists of all device nodes and the sink. Without loss of generality, let $t = v_0$ denote the sink, $\{v_1, \ldots, v_N\}$ denote ordinary nodes, and $\{v_{N+1}, \ldots, v_{N+M}\}$ represent supernodes. $E$ denotes the set of edges. Edge $(u, v), \{u, v\} \in V$, represents a communication link between node $u$ and $v$. For a source vertex $v_j$, each edge $(u, v) \in E$ associates a capacity $c_j(u, v) = 1$, a flow $f_j(u, v) \geq 0$ and a cost $a(u, v)$ ($P_{min} \leq a(u, v) = q(r_{u,v}) \leq P_{max}$), where $r_{u,v}$ denotes the Euclidean distance between nodes $u$ and $v$, $q(r_{u,v})$ is the predefined power consumption function, and $P_{min}$ and $P_{max}$ are the minimum and maximum power that device nodes can be assigned, respectively. $G$ is defined under the full load situation, i.e., an edge exists between two nodes if there exists a transmit power ($\leq P_{max}$) to maintain a communication between them.

Thus, the topology control with heterogeneous connectivity requirements (TCHCR) in the HSN can
be formulated as an optimization problem, which minimizes the total transmit power of the network and generates the network topology with minimized number of edges, subject to the connectivity constraints of all nodes. The formulated optimization problem is as follows.

\[
\text{minimize } F = \sum_{i=0}^{N+M} p_i \\
\text{minimize } \sum_{\{u,v\} \in E} h(u,v) \text{ when } F = F_{\text{min}}
\]

\[
\text{s.t. } \\
h(v_j,t) = 1 \text{ when } 1 \leq j \leq N \\
h(v_j,t) = k \text{ when } N + 1 \leq j \leq N + M \\
0 \leq f_j(u,v) \leq c_j(u,v) \\
\sum_{u:(v,u) \in E} f_j(u,v) = \sum_{u:(v,u) \in E} f_j(v,u) \quad \text{if } v \in V \setminus \{v_j,t\} \\
\sum_{u:(v,u) \in E} f_j(u,v) \leq 1 \text{ for all } v \in V \setminus \{v_j,t\} \\
\sum_{u:(v,u) \in E} f_j(v,u) \leq 1 \text{ for all } v \in V \setminus \{v_j,t\} \\
P_{\text{min}} \leq a(u,v) = g(r_{u,v}) \leq P_{\text{max}} \\
c_j(u,v) = 1, (u,v) \in E \\
f_j(u,v) = \{0,1\}, (u,v) \in E \\
a(u,v) \leq p_u, p_v \\
\text{if } \exists j, f_j(u,v) = 1, (u,v) \in E
\]

where \(h(u,v)\) denotes the total number of flows from source \(u\) to destination \(v\). Constraints (3) and (4) denote the connectivity requirements for ordinary nodes and supernodes, respectively; constraint (5) represents that the amount of the flows through an edge is no more than its capacity; constraint (6) represents flow balance for an intermediate node; constraints (7) and (8) mean that the outgoing and incoming flows at an intermediate node must be no more than 1, since paths are node-disjoint with each other; constraint (9) means the cost of sending a unit of flow through an edge should be no more than the maximum transmit power; constraints (10) and (11) represent that the amount of flows equals the number of paths; and constraint (12) represents that two incident nodes of one edge have sufficient transmit power to maintain the data transmission between them if the edge is selected.

**Theorem 1** The optimization problem defined in (1)-(12) is NP-hard.

**Proof.** See Appendix.

Due to the NP-hardness of the formulated optimization problem, for practical implementation, we need to search suboptimal solutions with low computational complexity.

### IV. Topology Adaptation Algorithm (TAA)

In this section, suboptimal solutions to the optimization problem defined in the previous section will be discussed. The main idea of our work is to first form a connected network to meet the connectivity requirements of all ordinary nodes, and then propose a connectivity augmentation algorithm to satisfy the requirement of \(k\)-connectivity of all supernodes. We name the combination of connected network formation and CAA as TAA.

#### A. Connected Network Formation

Given the network topology, we define a power assignment problem, which tries to form a connected network with the objective of minimizing the total power consumption. Unfortunately, it has been proved that constructing a minimum-power connected network in a 2-dimensional plane is NP-hard [30]. In order to reduce the computational complexity, in this paper, we use the concept of minimum weight spanning tree (MST) [24], which can form a connected network graph and can obtain two-approximation for the power assignment problem [25]. There are many algorithms in the literature for rapidly achieving a minimum-weight spanning tree. In this paper, the fast approximation algorithm in [24] is utilized. Let \(G'=(V, E')\) represent the spanning tree obtained, where the set of edges \(E'\) has a size of \(N + M\). Let \(p_j^{\text{init}}\) denote the transmit power of node \(v_j\) that can maintain all edges from it in \(E'\), i.e.,

\[
p_j^{\text{init}} = \max \{p(r_{v_j,u}) \text{ for all } u : (v_j, u) \in E'\}
\]

By using minimum weight spanning tree, the computational complexity is reduced to be less than \(O(m\log(N + M + 1)) \leq O(n^2 \log n)\), where \(m\) denotes the number of edges in the network when the network runs at the full load, and \(n = N + M + 1\) denotes the network size.

#### B. Connectivity Augmentation Algorithm (CAA)

Although a connected network has been formed, there is no guarantee that \(k\)-connectivity is met for all supernodes. Thus, some extra edges have to be augmented by assigning more power to some vertexes. Note that the edge augmentation procedure should keep the overall power consumption minimized.
We modify the graph $G$ by assigning each edge $(u,v)$ a weight $w_{u,v}$, which denotes the extra power needed if edge $(u,v)$ is augmented to the connected graph $G'$. Let the power assignments in $G'$ on node $u$ and $v$ are $p_u$ and $p_v$, respectively. Then, weight $w_{u,v}$ can be calculated as

$$w_{u,v} = |\min\{0, p_u-q(r_{u,v})\}| + |\min\{0, p_v-q(r_{u,v})\}|$$

In (14), bidirectional transmission is considered so that $w_{u,v}$ includes the power increment on both nodes $u$ and $v$.

The proposed CAA consists of following four main steps.

1) Based on the undirected graph $G = (V,E)$, for each supernode $v_j$ $(N + 1 \leq j \leq N + M)$, we calculate the maximum number of node-disjoint paths from it to the sink $t$ via graph transformation [26]. The detailed transformation procedure is shown as follows. i) For each node $v \in V \setminus \{t\}$, we replace it with two vertices, denoted as $v_{in}$ and $v_{out}$, which are connected by a directed edge $<v_{in}, v_{out}>$. ii) For each undirected edge $(u,v)$, $v \in V \setminus \{t\}$, we replace it with two directed edges, denoted as $<u_{out}, v_{in}>$ and $<v_{out}, u_{in}>$. If $v = t$, then we replace edge $(u,t)$ with one directed edge $<u_{out}, t>$. iii) Each directed edge is given capacity 1. After transformation, we derive a new digraph, denoted as $G_{split} = (V_{split}, E_{split})$. According to [26], edge disjoint paths of $G_{split}$ are identical to node-disjoint paths in $G$.

In order to better illustrate the transformation procedure, we consider an example as shown in Fig. 1. The original network topology is shown in Fig. 1(a), where node $A$ is a supernode, and $B, C, D$ are device nodes (each node may either be a supernode, or an ordinary node). In the figure, the weight of each edge denotes the Euclidean distance between two nodes, and the numbers associated with nodes denote the currently assigned transmit power. For the purpose of explanation, in the example, we adopt a simple power consumption model $q(r_{u,v}) = r_{u,v}^2$, and $p_{\min}$ and $P_{\max}$ are set to 0 and 25, respectively. Fig. 1(b) shows the connected graph $G'$ obtained by MST, where the initial power of nodes are assigned. Fig. 1(c) shows the derived digraph $G_{split}$ after the transformation procedure. Here, any vertex $u$ except the sink node $t$ is split into two vertices, namely $u_{in}$ and $u_{out}$. The numbers under arrows represent the capacity of edges, which equals 1. Obviously, in $G_{split}$, the total number of vertices $n' = 2(N + M) + 1$. The total number of edges in $G_{split}$, $m'$, can be calculated as follows. For any edge $(u,v)$ in $G$ ($v \neq t$), there are two corresponding edges in $G_{split}$. Hence, the total number of such edges are $2(m - n_{\text{neigh}}(t))$, where $m$ denotes the number of edges in $G$, and $n_{\text{neigh}}(t)$ denotes the number of $t$'s neighbors, where neighbors of a node is defined as nodes which can communicate directly with node $v$ under the transmit power constraint. For each edge $(u,v)$ in $G$ with $v = t$, the edge corresponds to one directed edge in $G_{split}$, i.e., the number of such edges is $n_{\text{neigh}}(t)$. Moreover, for each vertex $u$ in $G$, there is one directed edge from $u_{in}$ to $u_{out}$. Thus, in summary, we have $m' = 2m - n_{\text{neigh}}(t) + n$.

2) After graph transformation, the maximum number of feasible flows to the sink can be calculated. For each supernode $v$, finding the maximum number of node-disjoint paths to $t$ in $G$ is equivalent to finding the maximum number of flows between $v_{out}$ and $t$, which can be solved by algorithms such as Ford-Fulkerson algorithm [27], [28]. We define the output of the algorithm as $g(v,t)$ denoting the number of node-disjoint paths, and $T = \{t_1, t_2, \cdots, t_{g(v,t)}\}$ the set of these paths. In order to augment the connectivity of supernodes while minimizing extra power consumption, potential links with smaller weight should be added to graph $G'$, where the corresponding weight values can be calculated based on equation (14). Note that the weight values of edges in $G$ are variable rather than a constant, since any transmit power changes on one node may result in the changes of weights on all its adjacent edges. If an edge is selected to join in an edge disjoint path and its weight is more than 0, it means the power of incident nodes has to be increased by the edge's weight value. At the same time, the weight values of several edges with connection to these incident nodes need to be updated as well.

In Fig. 1(c), it is easy to find that for $A_{out}$, the maximum amount of flows to $t$ is 3, which is also the number of node-disjoint paths. In the example, according to the Ford-Fulkerson algorithm in [27], we have two flow paths, $\{(A_{out}, B_{in}), (B_{in}, B_{out}), (B_{out}, C_{in}), (C_{in}, C_{out}), (C_{out}, t)\}$, $\{(A_{out}, D_{in}), (D_{in}, D_{out}), (D_{out}, t)\}$, and one augmenting path $\{(A_{out}, C_{in}), (C_{in}, B_{out}), (B_{out}, t)\}$. In order to represent the paths by using vertices in $G$, recombi-
nation procedure is applied. In this procedure, we first examine every edge to determine whether its adverse edge exists on another path. If yes, both edges are removed. Then, the two involved paths exchange all edges after the removed ones in order to create two new paths. After that, the standard node-disjoint paths are formed, namely \(\{(A, B), (B, t)\}, \{(A, C), (C, t)\}\) and \(\{(A, D), (D, t)\}\), as shown in Fig. 1(d), where dashed lines represent the node-disjoint path, and the numbers on arrows derived by equation (14) denote the weights of corresponding edges.

3) If \(f(v, t) = k\), all paths in \(T\) have to be selected. Otherwise, we introduce a greedy strategy, greedy minimum extra-power-consumption path selection algorithm (GMEPS) to find the paths which introduce minimum extra power consumption. Due to variation on edge weights, GMEPS is implemented round by round. At round \(i\), let \(T^i = \{t'_1, t'_2, \ldots, t'_{f(v, t) - i + 1}\}\) denote the set of candidate paths and \(P^{e,i} = \{p^{e,i}_j, 0 \leq j \leq N + M\}\) denote the set of transmit power of device nodes after the \(e\)-th supernode carries out GMEPS at round \(i - 1\), where \(P^{1,i}\) is initialized to \(P^{\text{init}} = \{p^{\text{init}}_j, 0 \leq j \leq N + M\}\) as derived in subsection IV-A. Then, the path with minimum extra power consumption amongst \(T^i\), denoted by \(t'_{\text{num, index}}\), is selected at round \(i\). After that, the transmit power of nodes on the path and the weights of their adjacent edges are updated. Repeat the similar procedure \(k\) times to find \(k\) paths for each supernode \(v\), denoted by \(T^k_v\). The GMEPS algorithm is summarized in Algorithm 1.

We use node \(A\) in Fig. 1 as an example to CAA. At round 1, \(P^{A,1} = \{p^{A,1}_A = p^{A,1}_{A,1} = 16, p^{A,1}_B = p^{A,1}_{B,1} = 18, p^{A,1}_C = p^{A,1}_{C,1} = 16, p^{A,1}_D = p^{A,1}_{D,1} = 20, p^{A,1}_t = p^{A,1}_{t,1} = 20\}\). For path \(\{(A, C), (C, t)\}\), if it is selected, then nodes \(A, B\) and \(t\) need to increase their power to 16, 25 and 25, respectively, so that every edge on the path can be maintained. Hence the corresponding extra power consumption of the path can be calculated by \(16 - p^{A,1}_A + 25 - p^{A,1}_C + 25 - p^{A,1}_t = 14\). Following the similar way, at the current power settings, for paths \(\{(A, D), (D, t)\}\) and \(\{(A, B), (B, t)\}\), the total extra power consumptions are 0 and 8, respectively. Therefore, for supernode \(A\), path \(\{(A, B), (B, t)\}\) is picked first. The transmit power of nodes on the path will be updated to \(P^{A,2} = \{p^{A,2}_A = 16, p^{A,1}_{B,1} = 18, p^{A,1}_C = 16, p^{A,1}_D = 20, p^{A,1}_t = 20\}\), and the weights of their adjacent edges will be adjusted as well, as shown in Fig. 1(e). Similarly to the procedure in round 1, path \(\{(A, D), (D, t)\}\) is successively chosen as the second node disjoint path of supernode \(A\) to \(t\), where the corresponding extra power consumption is 8 as shown in Fig. 1(f). Obviously, the power increment in the augment procedure depends on the
Algorithm 1 GMEPS algorithm

1: input: \( T^i \), round \( i \), \( P^i \)
2: out: \( t'_{\text{num\_index}} \)
3: \( j = g(v,t) - i + 1 \) /* \( j \) denotes the number of paths in \( T^i \)
4: \( B_{i,j}=0 \) /* \( B_a \) indicates the total extra power consumption if path \( t'_a \) is selected, initialized to 0 */
5: \( S'_{i,j} = P^i \) /* \( S'_a \) indicates the current transmit power of nodes if path \( t'_a \) is selected, initialized to \( P^i \) */
6: for \( h = 1 \) to \( j \) do
7:     length=the hops of \( t'_h \)
8:     for for \( l = 1 \) to length do
9:         \( B(j) = B(j) + w_{el} \)
10:        according to Equation (12), adjust the power of its incident nodes \( \rightarrow S'_{h} \)
11:        according to Equation(13), adjust the weight values of their adjacent edges
12: end for
13: end for
14: \[ \text{min}_{num},\text{num\_index}]=\text{min}(B) \text{ } /* \text{num\_index} \]
15: \( P^{i+1} = S'_{\text{num\_index}} \)
16: adjust the weight values of edges according to \( S'_{\text{num\_index}} \)

order of path selection. Such property distinguishes our problem from the traditional minimum cost flow problem and makes existing solutions infeasible.

4) Merging edges. We can construct a new sparse network topology graph \( G'' = (V, E'') \), where \( E'' = \{e : e \in E'\text{of}G'\text{ and}c \in \bigcup_{u=V_{N+1}}^{V_{N+M}} T_k \} \). \( G'' \) is the final algorithm output, which can meet the connectivity requirements of all device nodes.

The pseudo-code of the proposed TAA algorithm is summarized in Algorithm 2

V. PREPROCESSING ALGORITHM (PA)

From simulation results as shown in Section VII, we find that many long edges are selected for path formation, which leads to high power consumption of incident nodes. By considering the fact that for a feasible network, deleting some long edges may not affect the connectivity of supernodes, we proposed a PA to prevent long edges from participating in the process of selection.

Algorithm 2 TAA algorithm

1: input: \( G = (V,E) \)
2: output: \( G'' = (V, E'') \)
3: use a minimum-weight spanning tree algorithm to form a connected network \( G'_1 = (V, E'_1) \)
4: for \( j = 1 \) to \( N + M \) do
5:     \( p_{\text{init}} = \max\{p(r_{v,u}) \text{ for all } u : (v_j, u) \in E'\} \)
6: end for
7: get a digraph \( G_{\text{split}} = (V_{\text{split}}, E_{\text{split}}) \) via graph transformation
8: for \( j = N + 1 \) to \( N + M \) do
9:     by a max flow algorithm, get the outputs \( T^1 = \{t_1, t_2, \cdots, t_{f(v_j)}\} \) for supernode \( v_j \)
10:    for \( i = 1 \) to \( k \) do
11:        implement GMEPS algorithm to get the path \( t'_{\text{num\_index}} \)
12:        \( T_{k_{v_j}} = T_{k_{v_j}} + \{t'_{\text{num\_index}}\} \)
13:        \( T_{i+1} = T_i \setminus \{t'_{\text{num\_index}}\} \)
14:    end for
15: end for
16: \( E'' = \{e : e \in E' \text{ in } G' \text{ or } e \in \bigcup_{u=V_{N+1}}^{V_{N+M}} T_k \} \)

Theorem 2 Consider a graph \( G(V, E) \), there exists a vertex \( t, t \in V \), an edge \( (u, v) \in E \), and \( g(u, v) \geq k + 1 \), where \( g(u, v) \) denotes the number of vertex-disjoint paths between \( u \) and \( v \) in the graph. Then, for any node \( v_1 \), if \( g(v_1, t) = k \) in \( G(V, E) \), we have \( g(v_1, t) = k \) in the graph \( G(V, E - (u, v)) \) as well.

Proof: Since \( g(v_1, t) = k \) in the graph \( G(V, E) \), there exist \( k \) vertex-disjoint paths for node \( v_1 \), denoted by \( P_s = \{p_1, p_2, \cdots, p_k\} \). Now we discuss the following three different cases.

1) If the edge \((u, v)\) is not on path \( P_s \), i.e., \((u, v) \notin P_s \), apparently, deleting \((u, v)\) does not affect the connectivity of \( v_1 \) to \( t \).

2) Consider the case that the edge \((u, v)\) is on one path \( p_k \in P_s \). Since \( g(v_1, t) = k \), according to [36], for a node-capacitated graph \( G(V, E) \), there exists a minimum vertex cut (Minimum vertex cut is defined as the smallest set of vertices in an undirected graph which separate two distinct vertices), denoted as \( S \), where \( v_1 \in S, t \in V \setminus S \), and the number of vertices in \( S \) is \( k \). Let \( A_1 \) and \( A_2 \) denote two connected subgraphs of \( G - S \) separated by the minimum vertex cut \( S \). Here \( G - S \) represents a new graph \((V_S, E_S)\), where \( V_S = V \), and \( E_S = E \) excluding whose are incident to nodes in \( S \). As illustrated
in Fig. 2, we assume that \( g(v_1, t) = k - 1 \) after removing the edge \((u, v)\). Then, the edge \((u, v)\) is a bottleneck edge for the connectivity between nodes \( v_1 \) and \( t \). Thus, \((u, v)\) contains a cut vertex \( u \), and \( S - u \), which consists of \( k - 1 \) vertices, is still the minimum vertex cut for nodes \( v_1 \) and \( v \). Since \( u \in A_1 \) and \( v \in A_2 \), \( u \) and \( v \) are disconnected so that \( \{S - u\} \) is also the cut for node pair \((u, v)\). Thus the number of node disjoint paths between \( u \) and \( v \) is \( \leq k - 1 \). It is contradict with the fact that \( g(u, v) = k + 1 \) and removing one edge \((u, v)\) can at most reduce \( g(u, v) \) by 1.

3) If the edge \((u, v)\) belongs to at least two disjoint paths, then it violates the assumption that any path in \( Ps \) is vertex-disjoint, so that this case does not exist.

In summary, the statement holds for all cases. ■

Theorem 2 indicates that for any two nodes \( u \) and \( v \), if there are more than \( k \) node-disjoint paths between them and the maximum required network connectivity for all nodes in the network is \( k \), deleting edge \((u, v)\) will not affect the network connectivity of all nodes. Since removing long edges may result in saving of network power consumption, we adopt Theorem 2 that unnecessary long edges can be deleted as many as possible.

Following this idea, we propose a PA. In PA, we check all edges in a descending order of Euclidean distances, whether their incident nodes are \( k+1\)-connected. If such edges exist, they will be deleted in orders till the number of edges connecting to any node is no more than \( k \). The pseudo-code of the proposed PA is summarized in Algorithm 3.

**Algorithm 3 PA**

1: input: \( k \) and \( G = (V, E) \), where the corresponding network of \( G \) runs at a full-load.
2: out: \( G = (V, E) \) where a part of edges in the original graph are deleted.
3: \( B_i = \) the neighbor number of node \( i \)
4: \( D_i = \) a list of node \( i \)'s neighbors
5: for \( j = 0 \) to \( N + M \) do
6: \( C_i = \) order \( D_i \) of size \( B_i \) according to the distance in descending order
7: end for
8: \( Max\_Num = Max(D_{1:N+M+1}) \)
9: \( h = Max\_Num \) to \( k + 1 \) do
10: for \( i = 0 \) to \( N + M \) do
11: if \( B_i \geq k + 1 \) then
12: if \( E(i, C_{i,h}) \neq 0 \) then
13: utilize Ford-Fulkerson algorithm to calculate \( f(i, C_{i,h}) \)
14: if \( f(i, C_{i,h}) \geq k + 1 \) then
15: \( E(i, C_{i,h}) = 0 \), \( B_i = B_i - 1 \)
16: \( E(C_{i,h}, i) = 0 \), \( B_{C_{i,h}} = B_{C_{i,h}} - 1 \)
17: end if
18: end if
19: end if
20: end for
21: end for

**VI. PERFORMANCE ANALYSIS OF THE PROPOSED ALGORITHM**

In this section, the performance of TAA is analyzed in terms of success rate and computational complexity. The analysis results are summarized in the following two theorems.

**Theorem 3** Given a feasible network, which meets all constraints (3)~(12), then our proposed algorithm will definitely produce a feasible solution, i.e., the success rate is 1.

**Proof:** By Theorem 1, the PA will not affect the connectivity of any node in the network. For a feasible network where the connectivity requirements of all device nodes can be satisfied, the minimum-weight spanning tree algorithm can guarantee the production of a connect graph. The augmentation
algorithm establishes the node-disjoint path based on the graph at a full load. For Ford-Fulkerson algorithm utilized in the augmentation algorithm, it has been proved that it can produce an optimal solution with the maximum number of paths [28]. It means that for supernodes, if it is $k$ vertex-connected to $t$ in the network under a full load, the augmentation algorithm can always find these paths.

**Theorem 4** The time complexity of the proposed algorithm is $O(n^3 \log^2 n)$, where $n$ is the number of nodes in the network.

*Proof:* The proposed algorithm consists of three sub-algorithms. They are a) the PA, b) the minimum-weight spanning tree algorithm, and c) the CAA. To analyze the time complexity of these sub-algorithms, we need to first find i) the average number of neighbors for any node in the network, ii) the time complexity of Ford-Fulkerson algorithm, iii) the length of paths given by the Ford-Fulkerson algorithm, and iv) the time complexity of the reverse transformation of graph $G_{\text{split}}$.

i) It is shown in [38] that $ae \log n = O(\log n)$ is the upper bound of average number of neighbors for forming a network in which any pair of nodes is $k$-connected. Here, $a$ denotes a real number larger than 1 and $e$ is the base of natural. It is known that when the capacity of edges is an integer, the time complexity of Ford-Fulkerson algorithm is $O(\|E\| \|f\|)$, where $\|E\|$ denotes the number of edges and $f$ is the maximum flow. Since the maximum flow of a node is impossible to exceed the number of the node’s neighbors, we have $f = O(\log n)$.

ii) From subsection IV-A, it is given $\|E\| = m' = 2m - n_{\text{neigh num}}(t) + n$. Thus, the time complexity of the Ford-Fulkerson algorithm is equal to $O(n \log^2 n)$ by considering $2m - n_{\text{neigh num}}(t) + n < 2n \log n + n = O(n \log n)$.

iii) In graph $G_{\text{split}} = (V_{\text{split}}, E_{\text{split}})$ with $\|V_{\text{split}}\| = n' = 2n - 1$ and $\|E_{\text{split}}\| = m' = 2m - n_{\text{neigh num}}(t) + n$, the length of any path obtained by the Ford-Fulkerson algorithm is no more than the number of nodes, and there exist no more than $O(\log n)$ paths. Therefore, the total number of edges on such paths is $O(n \log n)$.

iv) Within the recombination procedure of paths, for each of these edges on the first picked path, there are $(\log n' - 1)n'$ comparisons at the worst. Similarly, for each of these edges on the second path, the number of comparisons decreases to $(\log n' - 2)n'$. Thus, we induce that the time complexity of combining paths at the worst case is $T(n) = (\log n' - 1)n' + (\log n' - 2)n' + \cdots + (\log n' - 1)n' + \cdots + n' < n'^2 \log^2 n' = (2n - 1)^2 \log^2 (2n - 1) = O(n^2 \log^2 n)$.

We first analyze the time complexity of the PA as shown in Algorithm 3. It is not hard to find that step 13 is included in a double-loop and utilizes the Ford-Fulkerson algorithm to calculate $f(i, C_{i,h})$, which is the most time consuming operation. We consider an extreme case, where the algorithm has to be run for every edge in the graph. Therefore, the time complexity of the PA equals $O(n \log^2 n) \times m = O(n^2 \log^3 n)$. For the minimum-weight spanning algorithm, we have shown in subsection IV-A that it will take no more than $O(n^2 \log n)$ amortized time. The CAA consists of four steps. Compared to steps 2 and 3, the runtime of steps 1 and 4 is negligible. For each supernode, it is required to implement the Ford-Fulkerson algorithm, recombine overall paths and split vertices, and select $k$ best paths from all candidate paths. From the analysis above, steps 2 and 3 need $O(n \log^2 n)$ and $O(n^2 \log^2 n)$ time in complexity, respectively. In order to select one best path from candidate paths, it is required to calculate the corresponding extra power consumption of all candidate paths and compare the values with each other. Since the number of paths is $O(n)$, calculating the corresponding extra power consumption runs in $O(n)$. Moreover, the number of candidate paths at the beginning is $\log n$, and the number gradually decreases by 1 till $k$ best paths are selected. Thus the time complexity is given by $O(n) \log n + O(n)(\log n - 1) + \cdots + O(n)(\log n - k) < kO(n) \log n = O(n \log n)$. Considering an extreme case that all nodes are supernodes, we can derive the time complexity of the CAA equals $(O(n \log^2 n) + O(n^2 \log^2 n)) = O(n^3 \log^2 n)$. In summary, the overall time complexity of the proposed algorithm is $O(\log^2 n) + O(n^2 \log n) + O(n^2 \log^2 n) = O(n^3 \log^2 n)$.

**VII. Simulation Results**

In simulation, we consider a HSN which is deployed in a 100 m $\times$ 100 m area with total 90 device nodes and one sink. The number of supernodes $M$ varies from 2 to 20. The connectivity requirement for the supernodes, $k$, ranges from 2 to 4. The
maximum transmission range is fixed at 20 m. The same power consumption model in [29] is used, where the path loss exponent $\alpha$ is set to 2 and the transmission quality parameter $\beta$ is set to 1.

In order to highlight the effect of the proposed PA, we simulate separately the proposed TAA and TAA+PA. For comparison, two other algorithms are also simulated. One is Random connection Algorithm (RA), which randomly selects the edges within the maximum communication range for each node. Without loss of generality, the selection probability is set to 0.5. However, for other values, similar observations can be obtained. The other is HA from [21]. The performance metrics used in the simulation are listed as follows.

- The normalized total power consumption: this is the total power consumption of all device nodes normalized by the power consumption under the full-load configuration.
- The success rate: it is defined as the percentage of supernodes which successfully achieve $k$-connectivity.
- The average degree of nodes: it is defined as the average number of edges incident to nodes in the network. Smaller node degree means sparser network topology constructed. Smaller degree of nodes can be beneficial for decreasing communication interference and improving the network capacity.
- The average length of paths: it is defined as the average hops of paths from supernodes to the sink. From data transmission point of view, short paths are preferred in order to reduce the effects from possible link and node failures.

Fig. 3 shows the resultant topologies by four algorithms on a same network, with $M = 10$, $N = 80$ and $k = 3$. In the figure, the blue node, red nodes, and black nodes represent the sink, supernodes, and ordinary nodes, respectively. Blue lines, green lines, and yellow lines denote the first, the second, and the third node-disjoint paths for a certain supernode, respectively. From the figure, we can see that by TAA and TAA+PA, all supernodes are $k$-connected to the sink, and all ordinary node are 1-connected. However, for RA, some supernodes have less than $k$ node-disjoint paths to the sink. Moreover, isolated device nodes exist in RA, even if edges are much denser compared to TAA and TAA+PA (In this simulation, the number of edges in Fig. 3 (c) is close to $\frac{5}{3}$ of that in Fig. 3(a) and 2 of that in Fig. 3(b)). Among the four sub-figures, we can also find that the network topology formed by HA seems to be simplest in terms of the number of edges. However, it cannot guarantee connectivity for all supernodes. In addition, we can see that the average length of the disjoint paths found by TAA is shorter than that by TAA+PA, but the network topology induced by TAA is denser.

To further illustrate the effectiveness of the proposed algorithms, we demonstrate the success rate of four algorithms in Fig. 4, where $M$ varies from 2 to 20. The figure clearly shows that TAA and TAA+PA can guarantee unit success rate for all values of $k$, as proved by Theorem 3. The success rate of RA is much less than 1 and decrease as the increase of $k$. For HA, it demonstrates similar performance as RA when $k = 2$ and performs worst for larger $k$ values. It is because HA selects a large number of nodes with low transmit power in each round to form the path to the sink, which reduces the chances for path formation in the next round. When $k$ is fixed, the number of supernodes, $M$, has marginal effect on the success rate for both RA and HA. The reason is that in both algorithms device nodes are equally treated without taking into account node types.

Fig. 5 shows the comparison in terms of the normalized total power consumption. From the figure, we can observe that RA consumes much more power than all other three algorithms. HA can achieve the minimum power consumption. However, its very low success rate as indicated in the previous discussion makes it infeasible for practical implementation. With the increase of $M$, both TAA and TAA+PA request more power. It is because more edges have to be augmented for a larger $M$ so that corresponding nodes need more power for maintaining more active edges. Moreover, the power consumption of TAA+PA increases gradually compared to TAA, which clearly demonstrates the advantages of the PA in power consumption.

Fig. 6 shows the performance of four algorithms in terms of average degree of nodes. The smaller degree of nodes, the better the performance of the algorithm. Less degree of nodes implies less communication interference and easier maintenance. From the figure, we can observe that TAA+PA has the best performance. When $k = 2$, 3 and 4, TAA+PA achieved the minimum value in the average degree
Fig. 3. The network topologies constructed from four different algorithms on the same deployment of network nodes.

Fig. 4. Comparison of TAA, TAA+PA, RA, and HA in terms of success rate.
of nodes. Moreover, for TAA and TAA+PA, the average degree of nodes increases with the increase of both $k$ and $M$. It is because for larger $k$ and $M$, more edges have to be augmented to the network to meet the network connectivity requirements of supernodes. For TAA, the value rapidly increases as the number of supernodes increases. On the contrary, the value in TAA+PA increases very slightly. The reason is that cut edges are reserved at PA phase, which helps to lower the average degree of nodes, whereas TAA does not consider the different criticality of edges in augmenting network connectivity.

Fig. 7 compares the average path lengths of supernodes for four algorithms. From the figure, we can see that in TAA, supernodes have the shortest paths for any value of $k$, while TAA+PA achieves the longest path. It is reasonable since most of the long edges are deleted in PA. Removing the edges reduces the chances of establishing paths via small number of mediate nodes. We can see that the increase of the average path length of supernodes in RA and TAA is very small as $k$ increases, while for TAA+PA and HA, such value is significantly decreased.

In summary, we can conclude that 1) TAA and TAA+PA can guarantee a feasible solution to the topology control problem with heterogeneous connectivity requirements. 2) TAA+PA outperforms TAA in terms of power consumption and simplicity of network topology. 3) Both RA and HA are not able to address the topology control problem. For RA, it suffers from high power consumption, low success rate and dense network topology, while for TA, its most salient disadvantage is very low success rate.

VIII. CONCLUSIONS

In this paper, a novel topology control algorithm, called topology adaptation algorithm, is proposed for HSNs so that heterogeneous connectivity can be achieved for different devices. For facilitating the practical implementation, a connected network formation method and a connectivity augmentation
algorithm are proposed for ordinary nodes and supernodes, respectively. In addition, we propose a preprocessing algorithm to delete long edges in advance without affecting the network connectivity, to further decrease the total network power consumption and simplify the network topology. Simulation results show that TAA+PA has a better performance than TAA in general. It can form a sparse network with connectivity guarantee to both supernodes and ordinary nodes, and can significantly reduce the overall power consumption compared to the counterparts. For our future work, the constraint of path length will be integrated to our topology control problem.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61462021, 61262074, 6112008), Opening Project of Guangxi Key Laboratory of Trusted Software (Grant No. PF130549), Opening Project of Guangxi Experiment center of Information Science (Grant No. 20130318), and the Nature Science and Engineering Research Council of Canada (NSERC) Discovery Grant (Grant No. 341548).

APPENDIX A

Before proving the optimization problem TCHCR defined in (1)-(12) is NP-hard, we first introduce a simplified problem called the minimum edge problem with k-connectivity requirements with \( k \geq 2 \) to the sink (MECR) and prove it to be NP-hard.

**Lemma 1** The MECR problem is NP-hard.

**Proof:** Given any network graph \( G_1 = (V_1, E_1) \), it is known that the Hamiltonian cycle problem (HCP) in \( G_1 \) is a NP-hard problem. Suppose that \( H = (\hat{v}_1, \hat{v}_2, \cdots, v_{|V_1|}, \hat{v}_1) \) is a solution to the HCP of \( G_1 \), where \( \hat{v}_i \in V_1 \). We create an instance of MECR by letting \( t = v_1, \hat{V} = V_1 - t, \hat{E}_1 = \{(\hat{v}_1, \hat{v}_2), (\hat{v}_2, \hat{v}_3), \cdots, (v_{|V_1|-1}, v_{|V_1|}), (v_{|V_1|}, \hat{v}_1)\} \), and \( k = 2 \). We now show that \( G = (\hat{V}, \hat{E}_1) \) is a solution to the instance of MECR. It is obvious that for any vertex in \( \hat{V} \), it is on the Hamiltonian cycle path, and is 2-connected to \( t \). Since the minimum degree of any vertex 2-connected to \( r \) is 2, the extreme of the minimum number of total edges is \( 2 \times \|V_1\|/2 = |V_1| \), which is equal to \( \|E_1\| \). Lemma 1 is proved.

**Theorem 1** The TCHCR problem is NP-hard.

**Proof:** Let graph \( G_1 = (V_1, E_1) \) represent a network, where \( V_1 \) denotes all device nodes, \( t \) denotes the sink, and \( E_1 \) denotes the set of communication links between nodes. Suppose that \( H = (V_1, E_1) \) is a solution to the MECR problem of \( G_1 \), where the total number of edges selected from \( E_1 \) is minimal, and for any vertex \( u \in V_1 \), the connectivity between it and the sink satisfies the pre-specified requirements. First, we create an instance \( \hat{G}_1 = (\hat{V}_1, \hat{E}_1) \) of TCHCR. Let \( p_{\min} = P_{\max}, \hat{V}_1 = V_1, \hat{E}_1 = E_1 \). We can induce that \( H = (V_1, E_1) \) is a solution to the instance \( \hat{G}_1 \) of TCHCR. Obviously, for each vertex \( u \in \hat{V}_1 \), its respective connectivity is satisfied in \( H \). Since \( p_{\min} = P_{\max}, F_{\min} = \|V_1\|P_{\max} \). It keeps constant so that the value of objective function (1) can be seen as being minimized. As depicted above, \( \hat{E}_1 \) has a minimal number of edges, hence objective function (2) is satisfied as well. Theorem 1 is proved.
REFERENCES


