Int. J. Nonlinear Anal. Appl. 15 (2024) 5, 213–224
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2023.30488.4414

Extraction of disjoint paths in heterogeneous wireless sensor networks with mobile supernodes to enhance fault tolerance

Fariborz Ahmadi, Omid Abedi*, Sima Emadi

Department of Computer Science, Yazd Branch, Islamic Azad University, Yazd, Iran

(Communicated by Ehsan Kozegar)

Abstract

Heterogeneous wireless sensor networks with mobile supernodes consist of n sensors and m mobile supernodes. Disjoint paths are used in these networks to enhance fault tolerance, improve the network lifetime, and implement an effective load distribution. The network topology is disrupted because disjoint paths disappear when a supernode changes its location to improve the network lifetime and avoid the death of adjacent nodes. This paper proposes a distributed method for finding disjoint paths from ordinary sensors to mobile supernodes when supernodes move to new locations. The proposed algorithm will have a message complexity of $O(n^2\Delta)$ and an execution time of $O(n^2\Delta^2)$, in which n denotes the number of nodes, and Δ indicates the highest node degree. According to evaluation results, mobile supernodes led to a 96% longer lifetime than static supernodes, and the network fault tolerance with mobile supernodes was 7.1 times higher than the fault tolerance with static supernodes.

Keywords: disjoint paths, fault tolerance, heterogeneous wireless sensor networks, mobile supernodes, distributed algorithms
2020 MSC: 68M18

1 Introduction

Recent technological advances have led to the development of heterogeneous wireless sensor and actor networks, which can sense data from the environment, process data, and make data-based decisions. In these networks, nodes are responsible for sensing, whereas supernodes perform the decision-making operations and show specific decision-based responses. Moreover, supernodes have a high transmission range and a much higher battery capacity than nodes [1]. This paper assumes that the supernodes are interconnected, and data delivery to a supernode leads to data delivery to the sink node. Furthermore, supernodes have no energy limitations.

Due to the poor battery capacity of ordinary nodes in such networks, researchers have tried to design energy consumption reduction protocols. Load distribution and energy are major challenges in such networks. Under inefficient load distribution conditions, the nodes with larger loads are discharged earlier than others. The full energy discharge of a node eliminates the network paths and disconnects other nodes, shortening the network lifetime. According to earlier studies, an efficient load distribution could lead to a 30% lifetime improvement [26]. To balance the load and energy in a network, head nodes were replaced alternatively based on a set of parameters, e.g., the residual energy of nodes [8, 26].

*Corresponding author
Email addresses: fariborz84@gmail.com (Fariborz Ahmadi), oabedi@uk.ac.ir (Omid Abedi), emadi@iauyazd.ac.ir (Sima Emadi)

Received: January 2023 *Accepted:* May 2023

Supernode-neighboring nodes not only perform data sensing, processing, and transmission but also function as relays for other nodes. Therefore, such nodes have much higher loads than other nodes and are rapidly discharged. The mobile sink algorithms were proposed to handle load distribution and premature sink-adjacent node death challenges [14, 15, 16]. In these algorithms, the energy levels of network nodes are estimated through closed matrices and energy matrices to determine the next migration of each sink by using these estimates. The dominating set approximation algorithm with a complexity of $o(n^2)$ was employed to determine all the migration points. In this algorithm, each node is either a migration point or a direct neighbor to a migration point. Furthermore, each node stores the paths to each sink at each migration point, increasing the number of control messages to build the topology. The evaluation results of these algorithms have shown that mobile sinks outperformed static sinks in network lifetime. In the proposed method, mobile supernodes are employed to implement an efficient load distribution, improve the network lifetime, and avoid the premature death of supernode-neighboring nodes. Furthermore, the disjoint paths of each node are determined through the previously stored paths.

Fault tolerance and connectivity preservation are other major challenges of such networks. When a node failure arises from environmental factors or energy discharge, the paths for other nodes will be destroyed and disconnected them from the network. Earlier studies utilized k disjoint paths to connect each node to a set of supernodes [3, 10], A node failure would break a maximum of one path for each node, and the nodes could remain connected to the supernodes through the residual $k-1$ paths in the network. This approach has a fault tolerance of $k-1$ nodes. However, the stationarity of supernodes remains a daunting challenge, and the nodes at smaller distances from the supernodes are discharged earlier than the other nodes, disconnecting the distant nodes from the supernodes. Earlier studies not only extracted k disjoint paths but also used other stored paths to enhance fault tolerance [11]. Once a path fails due to a node failure, other paths will be recovered from the path information table to replace the failed paths. Furthermore, paths are determined with respect to the energy consumption and residual energy of nodes. Closer neighbors are allocated to nodes of lower energy. In this approach, supernodes are static, and nodes closer to supernodes are discharged earlier. However, the network has a longer lifetime and higher fault tolerance than the algorithm in [3] due to the stored path extraction. In the proposed method, the mobility of supernodes improves fault tolerance and lifetime.

To handle the foregoing challenges, the proposed methodology assumes constant supernode locations, and k disjoint paths to the supernode are extracted for each node, constructing a topology. The movement of supernodes to new migration nodes can change the topology, and the previous topology information is used to develop a new topology with new disjoint paths. This reduces the number of control messages to develop a topology.

This paper consists of different sections. Section 2 reviews the literature, whereas Section 3 describes the proposed methodology and algorithms. Section 4 reports and discusses the results. Finally, Section 5 draws a research conclusion.

2 Literature review

Topology control was used in earlier studies to reduce energy consumption in homogenous networks [7, 17, 18, 19, 24, 25]. The transmission power of networks was adjusted through the existing nodes at each moment of the network lifetime. In these methods, k disjoint paths to the sink were constructed for each node, incorporating energy consumption distribution and fault tolerance into the model.

Clustering was exploited for load distribution in networks [2, 4, 20]. For a set of nodes, a head node was determined with respect to different parameters such as the residual energy, directing sensed data through the head node to the sink. Since the head node would function as a relay for the other nodes and be discharged earlier, the head node was alternatively changed through nodes of higher energy as the head node.

Mobile sinks are an effective method for implementing an effective load distribution in networks. However, sink nodes randomly moved in the operational environment and distributed the load in the network [6]. Sinks would move to the next migration points based on parameters such as the residual of nodes and the number of steps from nodes to the sink node. According to the results, this approach outperformed stochastic methods.

A fault-tolerant technique was adopted to heterogeneous wireless sensor networks [3, 13], in which each node was connected through k disjoint nodes to a set of supernodes. This approach was employed to reduce the transmission range of nodes to improve the nodal lifetime and network lifetime. In this algorithm, nodes and supernodes are static, and nodes at smaller distances from the supernodes are rapidly discharged. The topology is built upon the transmission of control messages from supernodes to nodes, and the cost of each path is equal to the cost of the largest connection along the path.

The same method as that of [11] was employed in [3], except that the cost of each path was obtained with respect to the energy of the nodes in the path, selecting the lowest nodal energy of the path as the path cost. In this algorithm, apart from the extraction of k disjoint paths, other paths are placed in the path information table for the connectivity recovery phase so that the paths in the path information table could be used in place of the failed paths in the case of node or path failures. According to the results, this algorithm had higher connectivity than the one in [3] due to the involvement of nodal energy in path selection, lower transmission ranges of lower-energy nodes, and path information storage for the recovery phase [11].

The Kautz graph was employed to design a fault-tolerant low-energy network [22]. It is a d-degree k-diameter graph represented as $K(d, k)$. The Kautz graph indicates the number of lines that represent a certain path. It was demonstrated that a low degree of nodes reduced energy consumption and maintenance burden. However, a reduction in the node degree increases the network diameter and transmission delay. Hence, it is essential to establish a trade-off between the degree and diameter to maximize network topology performance. This method, however, had a disadvantage. In fact, all the paths are extracted statically, and global knowledge of the network is required as this method can be implemented in a centralized manner.

Earlier studies divided the operating domain of the network into sub-domains s that each node could be placed only in one sub-domain, and the nodes of each sub-domain would have nearly the same residual [12]. The domain would be re-divided when the nodes of a sub-domain were fewer than the threshold so that fault tolerance could be enhanced.

3 Disjoint path vector with mobile supernodes

The disjoint path vector with mobile supernodes (DPVMS) algorithm was intended to find k disjoint paths from each node to a set of supernodes at each migration point. First, it is assumed that supernodes are immobile, and the algorithm of topology construction and path information collection used in [3] was implemented at each node. It was used as the base algorithm in the present work. In this algorithm, a disjoint path table and a local path table were employed. First, each supernode transmits an "init" message, which includes the supernode ID, to its neighbors. Then, its neighboring nodes receiving the "init" message update the disjoint path table and transmit the "path-info" message, which contains the local node paths and their costs, to their neighbors. Once the path information has been received from the neighbors through the "path info" message, the current node transmits the paths in its local path table to its neighbors if the received information updates its disjoint path table. The disjoint path table is updated when the received paths have lower costs than the current paths or when a new path is added to the disjoint path table. The messages continue to be transmitted until the disjoint path tables of the nodes are no longer updated. Then, each node arranges its disjoint path table based on the path costs, selecting the first k paths as the connection paths of the corresponding node. Once k disjoint paths have been found, each node selects its neighbors — i.e., the next node in the path — and adjusts its transmission power to enable a direct connection to its farthest neighbor. Algorithms 1 and 2 [3] describe the pseudocodes of path information collection and disjoint path identification. Table 1 depicts the signs used in these algorithms. Each node establishes its links after finding neighbors and adjusting its transmission power and transmits an "advertisement" message to its connected supernodes. As a result, the relocation range of each supernode is determined to be the node points connected to the supernode.

Table 1: signs used in DPVMS

I	Received PathInfo message
k	Disjoint connectivity degree
T, T'	Set of local paths
D and D'	Set of disjoint paths
c and c'	Cost of disjoint paths
U	Union of two path sets
M	Total number of supernodes
Sr	Supernode ratio
N	Total number of sensorss
ODP	Old disjoint path
NDP	Updated disjoint path

Algorithm 1 Path information collection [11]

```
Input: I, L, k,
Output: D
    $T \leftarrow 0$;
    For all received path info message I do
      If I.Sender is a supernode then
        $r \leftarrow$ new path (I.Sender)
        If $r \notin T$ then
          $T \leftarrow T \bigcup r$
          Transmit pathinfo (T)
        End if
      else
        $D \leftarrow$ min-dis-set(T)      //Algorithm 2
         $C \leftarrow$ cost (D)
         $U \leftarrow I.T \bigcup T$;
         Sort (U)
         $T' \leftarrow \{pi\ \varepsilon U | i \leq L\}$
         $D' \leftarrow\leftarrow$ min-dis-set $(T')$
         $C' \leftarrow$ Cost $(D')$
         If $C' < C$ then
             $T \leftarrow T'$
             Transmit (T.Sid)
        End if
       End if
    End for
```

Algorithm 2 Finding disjoint path

```
Input: T and k
Output: D
    $D \leftarrow 0$;
     if $|T| > k$ then
       $Q \leftarrow \{q\varepsilon T||q| = k\}$;
        $c \leftarrow \infty$
        $q_{\min} \leftarrow 0$
        for all $q\varepsilon Q$ do
          if $q$ consists of disjoint paths then
           if $Cost(q) < c$ then
            $c \leftarrow Cost(q)$;
            $q_{\min} \leftarrow q$
           End if
         End if
       End for
       $D \leftarrow q_{\min}$;
    End if
```

3.1 Migration points for supernodes

This algorithm assumes that each supernode locates its sensor nodes through different technologies such as the global positioning system (GPS) or localization algorithms [18, 19, 24]. Since the nodes send an "advertisement" message to the connected supernodes, each supernode limits and determines its relocation range based on the locations of its associated nodes. As node relocation is aimed at enabling a balanced network load distribution between nodes, the nodes with more neighbors would be better supernode neighbors or migration nodes. In other words, for n sensor nodes with neighbor information in the network, the goal is to find q nodes through which all the nodes are directly available [14]. This is known as the dominant set and implies that each node is either in the dominant set or a direct neighbor to the nodes within the dominant set. In this NP-complete problem [9, 23], approximate algorithms with $O(n^2)$ complexity are employed to find the dominant set (i.e., migration points). When the migration points are found, a "migration point node" message is transmitted to those nodes, and the supernodes are relocated.

3.2 Disjoint paths at new migration points

When initial topology construction and migration point selection are completed, the supernodes begin to move between the migration points. Earlier studies employed the EL parameter, and the supernodes would start moving to the next migration point once EL% of the energy of the nodes in the adjacency of the migration was consumed [16]. The supernode moved to the next migration point at $t_{\min}$ [5]. This paper determined the next migration point by using the residual of the nodes adjacent to the migration point. The next migration point is the one whose neighbors have higher energy than the neighboring nodes of other migration points. This study adopted the methodology introduced in [16].

The DPVMS utilizes the disjoint path information of the initial topology and determines k disjoint paths for the next migration point of each node. When the network operation begins, each supernode moves to the new migration point and sends a "new location" message containing its new location and ID to its neighbors. The one-step neighbors of the supernode receive the message and form a direct path to the supernode, sending an "actor neighbor" message, including ID, new disjoint path (of the node and supernode), and the previous disjoint paths. The node receiving this message will be in three scenarios to find its disjoint paths:

1. The receiver checks its disjoint path table in the paths ending in the ID of the supernode to find the ID of the transmitter. If any, there will be new paths from the current node to the transmitter. For example, if supernode A transmits the "new location" message, the "actor neighbor" message is transmitted from node y to node w, and there is the path $w \to z \to y \to x \to A$ in node w, the new disjoint path in node w for the new migration point of supernode A is $w \to z \to y \to A$. As each path is a disjoint path, this algorithm ensures that new paths are also disjoint paths since each subset of a disjoint path is a disjoint set. An "actor neighbor" message may be transmitted from other nodes to node w; upon the reception of this message, node w determines the unions of the disjoint paths that have been found and the disjoint paths that have been found by the other nodes. Therefore, once transmitter y has received the message, the list of new disjoint paths for node w is $w_{\text{new disjoint path}} = w \to z \to y \to A$. Each node transmits the "actor neighbor" message to its neighbors after the disjoint path table has been updated.
2. The ID of the transmitter is not found in the disjoint path tale of the receiver. In such a case, the current node checks the disjoint path table of the transmitter to find its ID. If the current node finds its ID, it cuts and then reverses the path. For example, supernode A transmits the "new location" message, node y transmits the "actor neighbor" message to node w, node y is not present in the disjoint paths of node w, and path $y \to z \to w \to A$ in its previous disjoint path table. Then, the distance from the beginning of the path to the current node is cut and reversed; i.e., $w \to z \to y \to A$ is added to the disjoint path table of node w. Hence, the disjoint path list of node w is $w_{\text{new disjoint path}} = w \to z \to y \to A$. Since the path is disjoint for node y, there is a maximum of one path containing node w. When the disjoint path is updated, node w transmits the "actor neighbor" message.
3. The ID of the transmitter is not found in the disjoint path table of the receiver, and the ID of the node is not found in the previous disjoint path table of the transmitter. In this case, the current node extracts the paths in the new disjoint path table of the transmitter and transmits its own ID to the beginning of the path, building the disjoint paths of the new location. For example, supernode A transmits the "new location" message, node y transmits the "actor neighbor" message to node w, and the new path of node y is $y \to x \to A$. Then, $w_{\text{new disjoint path}} = w \to y \to x \to A$. When the disjoint path table is updated, the "actor neighbor" message is transmitted by the node.

These messages are transmitted until the disjoint paths of the nodes remain unchanged. When the paths are extracted by the nodes, Algorithm 2 is executed to check whether the paths are disjoint, selecting the first k paths with the lowest costs as the k disjoint paths. Then, each node adjusts its transmission range to achieve its k neighbors. Algorithm 3 represents the pseudocode for this procedure.

Figure 1(a) depicts five nodes and one supernode. Let k be 1, i.e., each node is connected to the supernode through one path. Let nodes e and c be migration points, and supernode A is initially adjacent to node e. The disjoint path of each node extracted by Algorithm 1 is shown next to the node. When the supernode moves to the next migration point (node c), the previous disjoint paths are not the correct paths to reach supernode A, and supernode A transmits a "new location" message to its neighboring node, i.e., node c (Figure 1(a)). Node c builds a disjoint path to supernode A and transmits an "actor neighbor" message to its neighbors, i.e., nodes d, b, and f. Node b checks its previous disjoint path, as it contains node c, the new path is $b \to c \to A$ (scenario 1). Node f also contains node c in its previous disjoint path and changes its path into $f \to c \to A$, as with node b (scenario 1). Node d checks its disjoint path; as it does not contain node c, node d checks the previous disjoint path received from node c to find its ID. This path contains node d and is then reversed into $d \to c \to A$ as the new disjoint path of node d (scenario 2). Then, node d transmits the "actor neighbor" message to its neighbor, i.e., node e. Here, the disjoint path of node e does not contain supernode d, and the previous disjoint path of node d does not include node e. As a result, node e extracts the new path of node d and adds its ID, building the new path of $e \to d \to c \to A$ (Scenario 3).

3.3 Message complexity and time complexity in DPVMS

According to earlier studies, the DPV algorithm had a message complexity of $O(n\Delta)$ and a time complexity of $O(n\Delta^2)$ [3]. DPV is the base algorithm of the present work, and Algorithm 3 with the same complexity as that of the DPV algorithm is executed at each migration point. For an equal distribution of nodes to supernodes, the number of nodes connected to each supernode is n/m, in which n is the number of nodes, whereas m is the number of supernodes. The results proved that the number of migration points in a graph with n nodes was $3/8n$ [21]. Hence, the total number of migration points in this algorithm was $m * \frac{3n}{8m} = \frac{3n}{8} = O(n)$. For the worst-case scenario, Algorithm 1 was executed at each migration point. Thus, the message complexity and time complexity of DPVMS were $O(n^2\Delta)$ and $O(n^2\Delta^2)$, respectively.

Algorithm 3 determines k disjoint paths for the next migration point

```
Input: I, L, k, ODP, NDP
Output: D
        $T \leftarrow 0$;
        For all received message I do
          If I is a new-location then
1: this message is submitted from supernodes
            $r \leftarrow$ new path (Sensor.ID $\bigcup$ I.ID)
            Sensor.NDP $\leftarrow r$
          else
            If I.ID $\varepsilon$ Sensor.ODP then
              $r \leftarrow$ new path (Sensor.ID to I.ID)$\bigcup$ I.NDP
              Sensor.NDP $\leftarrow r$
            End if
            If I.ID$\notin$ Sensor.ODP and Sensor.ID$\varepsilon$ I.ODP then
              $r \leftarrow$ new path (Sensor.ID down to I.ID)$\bigcup$ I.NDP
              Sensor.NDP $\leftarrow r$
            End if
            If I.ID$\notin$ Sensor.ODP and Sensor.ID$\notin$ I.ODP then
              $r \leftarrow$ new path (Sensor.ID $\bigcup$ I.NDP)
              Sensor.NDP $\leftarrow r$
            End if
           End if
           $D \leftarrow$ min-dis-set(Sensor.NDP)
           If $D \neq \emptyset$ then
             Transmit actor-nighbor (ID, ODP, NDP)
           End if
        End for
```

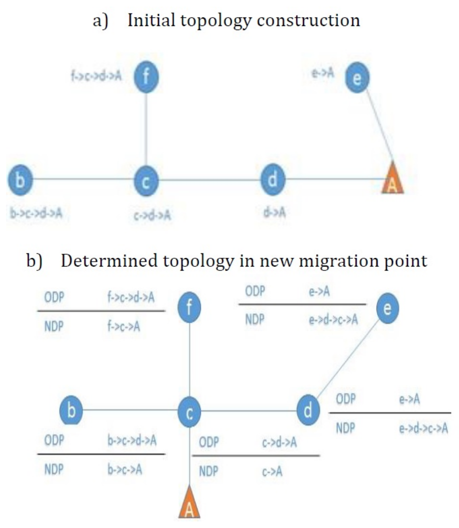

Figure 1: determining disjoint paths in DPVMS

4 Discussion

For evaluation, a comparison was drawn between DPVMS and DPV. The DPVMS was implemented in a custom simulator. It executed the algorithms on the proposed topology, calculating and reporting different performance criteria. The nodes and supernodes were assumed to be distributed uniformly and randomly across a 600 × 600mm area with an initial transmission range of 100m [3, 11]. The algorithms were executed for {100, 150, 200, 250, 300, 350, 400, 450, 500, and 550}, k=2, 3, and 4, and supernode ratios of 3% and 5%. The three algorithms were executed ten times for each experiment, reporting the average outputs.

Figure 2 depicts fault tolerance versus node failure for DPV and DPVMS. Performance is evaluated in the percentage of nodes disconnected from the network when all the nodes are connected to the network, despite the presence of failed nodes. If a node is disconnected from all the supernodes, supernode disconnection occurs for the node. According to the results, the proposed algorithm was effective in maintaining supernode connectivity. As shown in Figure 2, the DPV algorithm did not have effective fault tolerance due to its static system and the premature death of supernode-adjacent nodes. The death of nodes in the adjacency of supernodes disconnected other nodes from the supernodes, and the DPV algorithm could not cope with this challenge. The DPVMS algorithm handled the death of supernode-adjacent nodes through mobile supernodes. Therefore, DPVMS has higher fault tolerance than DPV. In DPMVS, fault tolerance increases when k increases, as more paths to the supernode are extracted. For example, for 550 nodes, a supernode ratio of 5%, and k=2, the node death rate for network disconnection is 52% in DPVMS, whereas a rise in k to 3 and 4 increases the node rate to 60% and 67%, respectively. Moreover, an increase in k raises the transmission efficiency, thereby shortening the network lifetime (Figure 3). Furthermore, an increase in the supernode ratio increases fault tolerance, as the transmission range and nodal energy consumption decline. For example, for 550 nodes, k=4, and a supernode ratio of 3%, fault tolerance was calculated to be 56%, whereas a rise in the supernode ratio to 5% increased fault tolerance to 67%. An increase in the number of nodes led to the extraction of more disjoint paths; therefore, fault tolerance was higher in a network with a larger number of nodes. For example, fault tolerance was 67% for 550 nodes, k=4m and a supernode ratio of 5%, whereas a reduction in the number of nodes to 100 decreased fault tolerance to 26%. The fault tolerance of the proposed DMPVS algorithm was 7.1 times higher than that of the DPV algorithm on average.

Figure 3 compares the samples (Figure 2) in the network lifetime. Accordingly, the number of nodes had no effects on the network lifetime in DPV, and nodes adjacent to supernodes were discharged earlier, leading to a shorter battery life. For DPVMS, the nodes adjacent to mobile supernodes, which are network hotspots, changed upon supernode relocation. The DPVMS algorithm strikes a more effective balance in the load and leads to a longer network lifetime. As mentioned, a rise in k increases the transmission range and reduces the node and network lifetimes in both DPV and DPVMS. For DPVMS at a supernode ratio of 5%, the average lifetime at k=2 was 1.15 and 1.25 times as long as the lifetime at k=3 and k=4, respectively. An increase in the supernode ratio reduced nodal energy consumption, extending network lifetime in both DPV and DPVMS. An increase in the number of nodes in DPVMS leads to the extraction of more disjoint paths, more effective load balancing, and a longer network lifetime. For DPV, however, the node density has no effects on the network lifetime due to the energy discharge of supernode-adjacent nodes. The lifetime of DPMVS was reported to be 1.96 times longer than that of DPV on average.

Figure 4 compares the total number of control messages transmitted in DPV and DPVMS at EL= 30%. The total number of these messages represents energy consumption in topology construction. Energy consumption in topology construction is lower when there are fewer messages.

According to Figure 4, the number of control messages increased with the number of nodes in both DPV and DPVMS. An increase in the number of nodes increased supernodes, and the movement of supernodes in DPVMS led to more control messages. Moreover, a rise in k increased the number of control messages, as more messages should be transmitted to extract the paths. The supernode ratio, however, had no significant effects on the number of control messages, for an increase in the supernode ratio reduced the number of migration points, with the number of supernode movements remaining nearly unchanged. According to Figure 4, the number of control messages in DPVMS was 5.1 times larger than that in DPV on average for k=2; DPVMS had 6.9 and 7.6 times as many control messages as DPV at k=3 and k=4, respectively. These additional messages were transmitted to inform nodes of supernode movements, construct the topology based on the current supernode locations, recover the supernode connectivity, and optimally use the resources and nodes. Furthermore, these messages play a key role in balancing the load, avoiding the premature death of nodes, and improving network lifetime.

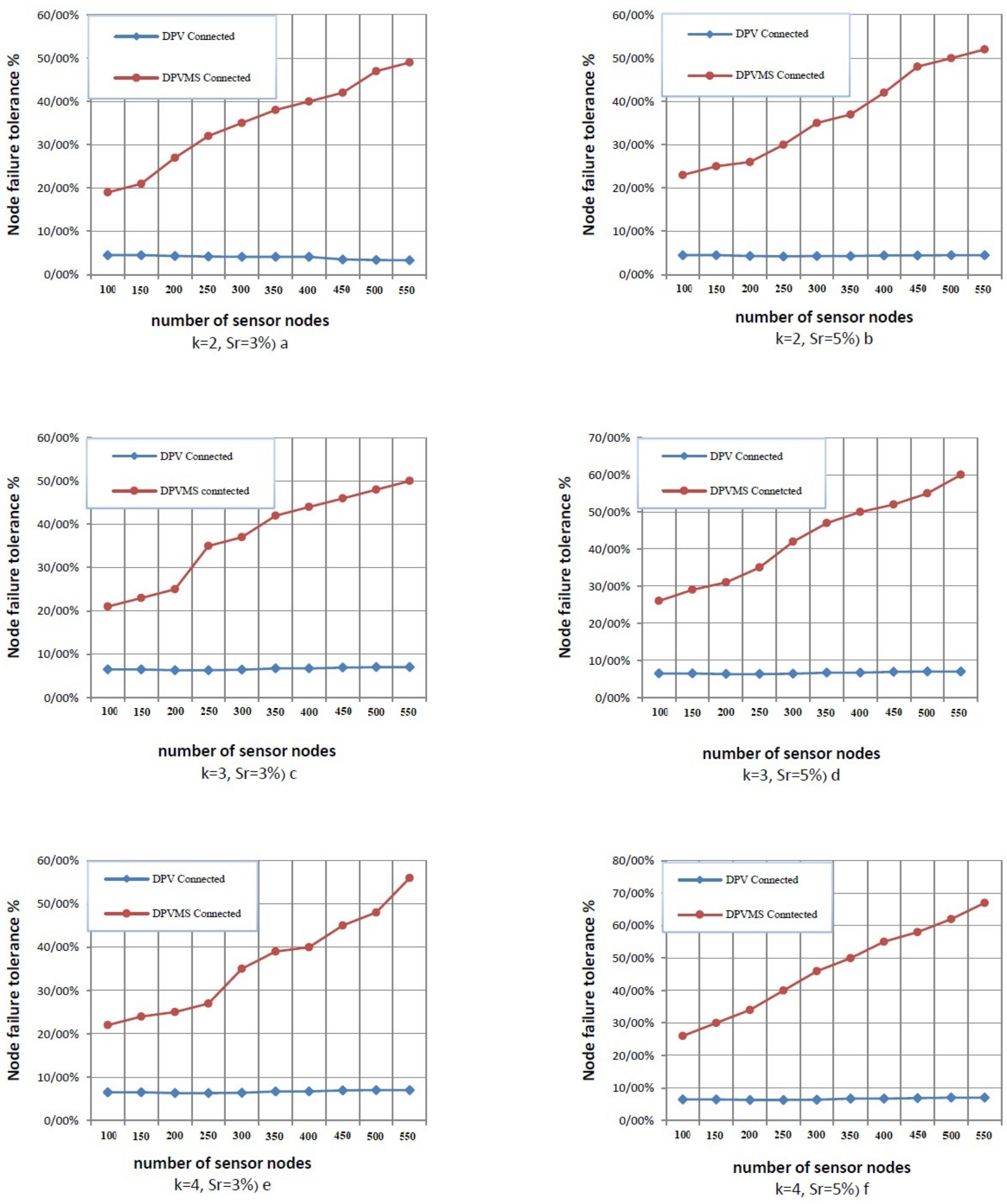

Figure 2: Percentage of failed nodes when supernode disconnection occurs

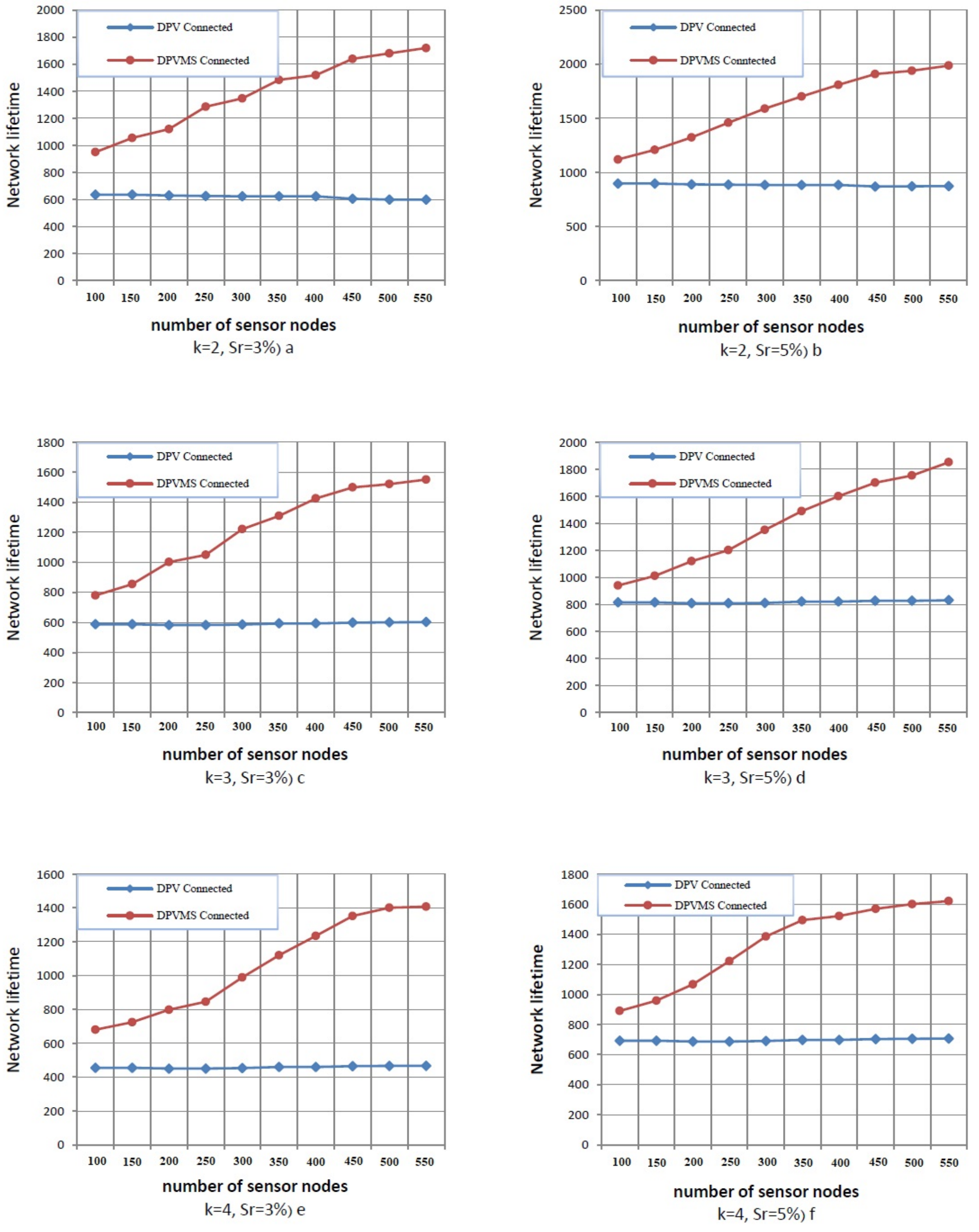

Figure 3: Network lifetime

Figure 5 draws a comparison between DPV and DPVMS in the number of connectivity recoveries for k=2, 3, and 4. The DPV algorithm is static and has no connectivity recovery, and a node failure can eliminate the path. The DPVMS algorithm, however, has mobile supernodes, and the nodes connected to supernodes recover their connectivity upon supernode movements through message transmission, which in turn leads to more transmitted messages in the DPVMS algorithm than in the DPV algorithm. Moreover, a rise in k raises the connectivity recovery frequency, as more nodes would correspond to supernodes. According to Figure 5, the proposed DPVMS algorithm had 5.1 times as many control messages as DPV for k=2, with a connectivity recovery frequency of 1170. The frequencies of connectivity recovery were 1246 and 1362 at k=3 and k=4, respectively.

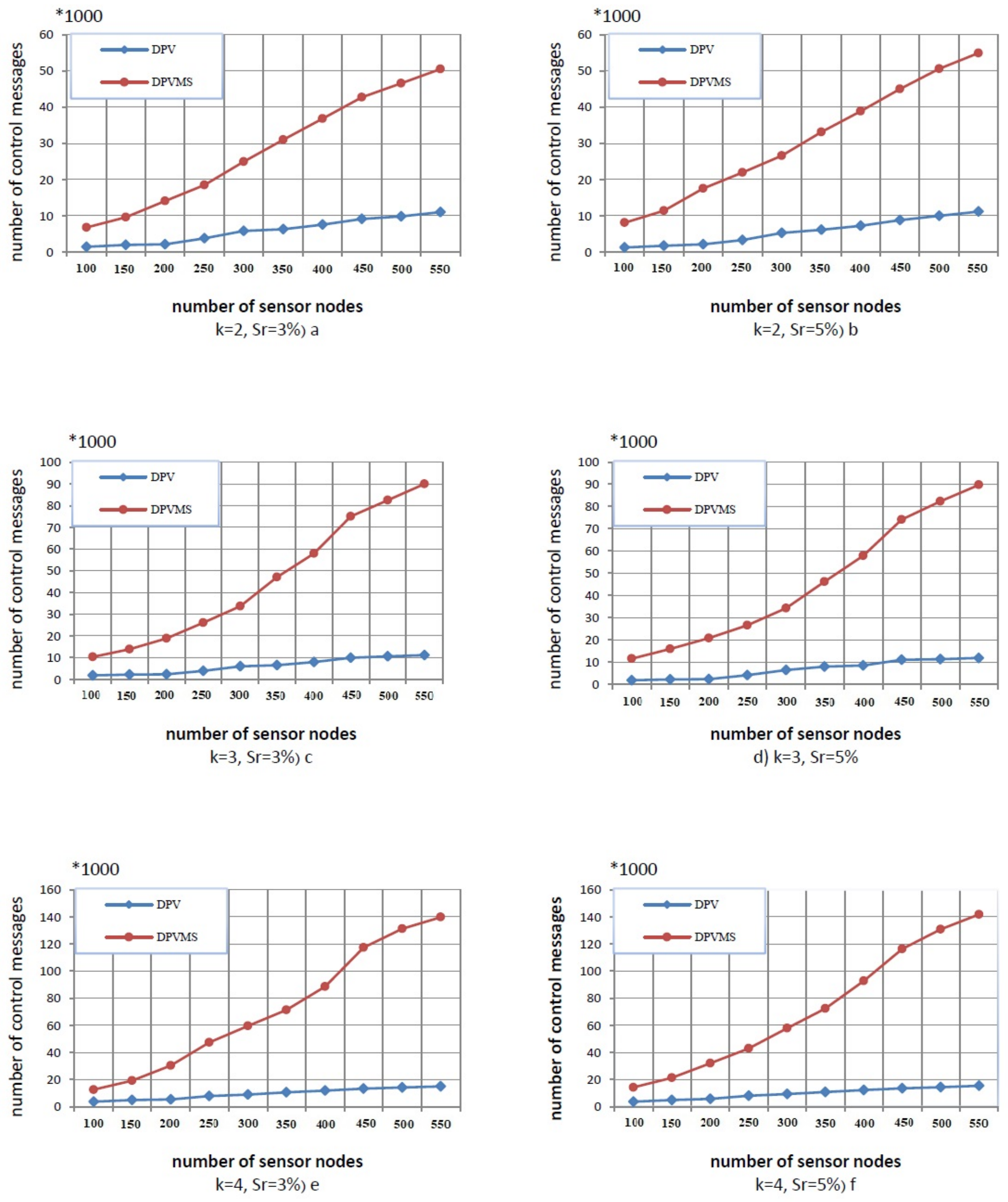

Figure 4: total number of control messages transmitted in DPV and DPVMS

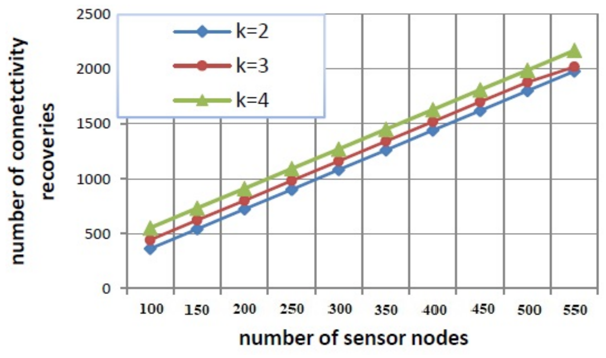

Figure 5: total number of connectivity recoveries

5 Conclusion

This paper proposed a distributed method for enhancing fault tolerance and improving the network lifetime by using mobile supernodes in heterogeneous wireless sensor networks. In the proposed DPVMS, each node is connected to a set of supernodes via disjoint paths. When a supernode migrates, the disjoint paths of each node become invalid, and the nodes extract k new disjoint paths via the previous paths. Mobile supernodes balance the load distribution in the network and extend nodal lifetime. The lifetime and fault tolerance of DPVMS were reported 1.96 and 7.1 times better than those of DPV, respectively. This paper used all the supernodes for data transmission, whereas the optimal number of supernodes to activate network connectivity and deactivate unnecessary supernodes can further improve the network lifetime and enhance fault tolerance.

References

[1] I.F. Akyildiz and I.H. Kasimoglu, *Wireless sensor and actor networks: research challenges*, Ad Hoc Networks **2** (2004), 351–367.

[2] M. Azharuddin, P. Kuila and P.K. Jana, *A distributed fault tolerant algorithm for wireless sensor networks*, Proc. IEEE Int. Conf. Adv. Comput. Commun. Inform., 2013, pp. 997–1002.

[3] H. Bagci, I. Korpeoglu and A. Yazici, *A distributed fault-tolerant topology control algorithm for heterogeneous wireless sensor networks*, IEEE Trans. Parallel Distrib. Syst. **26** (2015), no. 4, 914–923.

[4] S. Bandyopadhyay and E.J. Coyle, *An energy efficient hierarchical clustering algorithm for wireless sensor networks*, Proc. 22nd Annu. Joint Conf. IEEE Comput. Commun. **3** (2003), 1713–1723.

[5] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli and Z.M. Wang, *Controlled sink mobility for prolonging wireless sensor networks lifetime*, Wireless Networks **14** (2008), no. 6, 831–858.

[6] S. Basagni, A. Carosi and C. Petrioli, *Controlled vs. uncontrolled mobility in wireless sensor networks: some performance insights*, IEEE 66th Vehicular Technol. Conf., 2007, pp. 269–273.

[7] D.M. Blough, M. Leoncini, G. Resta and P. Santi, *The k-neigh protocol for symmetric topology control in ad hoc networks*, Proc. 4th ACM Int. Symp. Mobile ad hoc Netw. Comput., 2003, pp. 152–158.

[8] S.S. Chauhan and M.M. Gore, *Balancing energy consumption across network for maximizing lifetime in cluster-based wireless sensor network*, CSI Trans. ICT **3** (2015), 83–90.

[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, The MIT Press, 2nd eds, 2001.

[10] F. Deniz, H. Bagci, I. Korpeoglu and A. Yazici, *Energy-efficient and fault-tolerant drone-BS placement in heterogeneous wireless sensor networks*, Wireless Networks **27** (2021), 825–838.

[11] F. Deniz, H. Bagci, I. Korpeoglu and A. Yazõcõ, *An adaptive, energy-aware and distributed fault-tolerant topology-control algorithm for heterogeneous wireless sensor networks*, Ad Hoc Networks **44** (2016), 104–117.

[12] S. Henna, *Energy efficient fault tolerant coverage in wireless sensor networks*, J. Sensors **4** (2017), 1–11.

[13] V. Khalilpour Akram, Z. Akusta Dagdeviren, O. Dagdeviren and M. Challenger, *PINC: Pickup non-critical node based k-connectivity restoration in wireless sensor networks*, Sensors **21** (2021), no. 19, 6418.

[14] M. Koc and I. Korpeoglu, *Controlled sink mobility algorithms for wireless sensor networks*, Int. J. Distrib. Sensor Networks **10** (2014), no. 4, 167508.

[15] M. Koc and I. Korpeoglu, *Coordinated movement of multiple mobile sinks in a wireless sensor network for improved lifetime*, EURASIP J. Wireless Commun. Network. **2015** (2015), no. 1, 1–16.

[16] M. Koc and I. Korpeoglu, *Traffic-and energy-load-based sink mobility algorithms for wireless sensor networks*, Int. J. Sensor Networks **23** (2017), no. 14, 211–221.

[17] 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. **13** (2005), no. 1, 147–159.

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

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

[20] K.K. Mamidisetty, M.J. Ferrara and S. Sastry, *Systematic selection of cluster heads for data collection*, J. Netw. Comput. Appl. **35** (2012), 1548–1558.

[21] A. Poghosyan, *The probabilistic method for upper bound in domination theory*, PhD thesis, University of the West of England, 2010.

[22] H. Shen and Z. Li, *A Kautz-based wireless sensor and actuator network for real-time, fault-tolerant and energy-efficient transmission*, IEEE Trans. Mobile Comput. **15** (2016), no. 1, 1–16.

[23] S. Skiena, *The Algorithm Design Manual*, Springer, 2nd edition, 2008.

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

[25] 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*, Proc. IEEE Wireless Commun. Netw. Conf., 2013, pp. 1009–1014.

[26] Z. Xu, L. Chen, T. Liu, L. Cao and C. Chen, *Balancing energy consumption with hybrid clustering and routing strategy in wireless sensor networks*, Sensors **15** (2015), 583–605.