International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 6, Number 1, pages 107–123, January 2016

Topology Management for Reducing Energy Consumption and Tolerating Failures in Wireless Sensor Networks

Qian Zhao and Yukikazu Nakamoto
Graduate School of Applied Informatics, University of Hyogo,
Computational Science Center Building 702, 7-1-28 Minamimachi, Chuo-ku, Kobe 650-0047, Japan

Received: February 14, 2015
Revised: May 4, 2015
Revised: July 5, 2015
Accepted: August 12, 2015
Communicated by Chisa Takano

Abstract

We investigated energy efficient and fault tolerant topologies for wireless sensor networks (WSNs), addressing the need to minimize communication distances because the energy used for communication is proportional to the 2nd to 6th power of the distance. We also investigated the energy hole phenomenon, in which non-uniform energy usage among nodes causes non-uniform lifetimes. This, in turn, increases the communication distances and results in a premature shutdown of the entire network. Because some sensor nodes in a WSN may be unreliable, it must be tolerant to faults. A routing algorithm called the "energy hole aware energy efficient communication routing algorithm" (EHAEC) was previously proposed. It solves the energy hole problem to the maximum extent possible while minimizing the amount of energy used for communication, by generating an energy efficient spanning tree. In this paper, we propose two provisioned fault tolerance algorithms: EHAEC for one-fault tolerance (EHAEC-1FT) and the active spare selecting algorithm (ASSA). EHAEC-1FT is a variation of EHAEC. It identifies redundant communication routes using the EHAEC tree and guarantees 2-connectivity (i.e., tolerates the failure of one node). The ASSA attempts to find active spare nodes for critical nodes. It uses two impact factors, α and β, which can be adjusted so that the result is either more fault tolerant or energy efficient. The spare nodes fix failures by replacing them. In our simulations, EHAEC was 3.4 to 4.8 times more energy efficient than direct data transmission, and thus extended the WSN lifetime. EHAEC-1FT outperformed EHAEC in terms of energy efficiency when fault tolerance was the most important, and a fault tolerant redundancy was created when or before a failure occurred. Moreover, we demonstrated that the ASSA was more energy efficient than EHAEC-1FT, and the effect of using different α and β.

Keywords: Wireless sensor networks, Energy efficient, Fault tolerance, Routing algorithms

1 Introduction

Wireless sensor nodes are now extensively used in many applications to improve people's lives and for security. For example, they are used to manage air conditioners, lights, and alarms, to save electric power, and to warn people of danger by sensing environmental conditions. Sensor nodes are used for infrastructure elements such as bridges and tunnels to detect conditions that could lead to

problems such as a collapse, by sensing reflected sound waves. Wireless sensor networks (WSNs) are fast becoming ubiquitous. However, most sensor nodes in WSNs are battery powered, so we must consider the battery life because if the nodes are located in dangerous and/or difficult-to-reach places, they are practically impossible to maintain. Therefore, sensor nodes should be able to operate for prolonged periods without the need for a replacement battery, with the aim of maintenance-free operation.

The electric power used by a WSN is mainly consumed by the wireless communications, the sensors, and the CPU running the programs that control the sensors. The wireless communication consumes the most power [5] (up to 80% of the total power [6]). Therefore, the most effective way to improve the energy efficiency is to minimize the energy used for wireless communications. In wireless communications, energy consumption is mainly determined by the communication distance and the amount of transmitted data. In particular, because the energy consumption grows in proportion to the 2nd to 6th power of the communication distance [4][7], energy efficient data routing is needed to avoid the extra energy consumption caused by long communication distances. Another consideration is the "energy hole phenomenon" [4], where a node that consumes more energy than the other nodes because it relays more data will run out of power sooner. This non-uniform energy consumption results in longer communication distances, resulting in a shorter lifetime for the entire network. The energy hole problem can directly affect the energy efficient communication route. An energy hole aware energy efficient communication routing algorithm (EHAEC) that solves this problem was proposed in our previous work [1]. However, there are usually some unreliable sensor nodes in a WSN, so a fault tolerance is required when optimizing the communication topology. We propose two kinds of provisioned tolerance algorithms that tolerate node failures. EHAEC-1FT (EHAEC for one-fault tolerance) ensures that the network connectivity is preserved, and can tolerate one node failure in the network. EHAEC-1FT can be extended to X-nFT, which can be used by an arbitrary spanning tree routing algorithm, X, for n-fault tolerance. The other provisioned tolerance algorithm is the active spare selecting algorithm (ASSA), which attempts to designate backups for critical nodes. Compared with including redundant nodes that accompany the critical nodes, ASSA finds active spare nodes that can replace failed nodes. This paper is an extended version of [2], which was presented at the CANDAR 2014 workshop.

The remainder of this paper is as follows. We discuss some related studies in Section 2. Section 3 contains some preliminaries and a description of the EHAEC algorithm [1]. Sections 4 and 5 present the fault tolerance algorithms, EHAEC-1FT and ASSA. Section 6 contains our simulation results, and a discussion. Section 7 concludes the paper with a summary of the key points.

2 Related Studies

A number of studies have attempted to construct energy efficient wireless communication routes, solve the energy hole problem, and enhance fault tolerance in data communications.

Some studies attempted to extend the lifetime of a WSN by minimizing data communication distances through the use of protocols such as LEACH [8] and PEGASIS [9]. LEACH reduces the energy consumption of the nodes by forming clusters. The node selected as the cluster head collects each node's data, aggregates them, and sends the aggregated data directly to the cluster base station. PEGASIS improves on LEACH by constructing a chain route in which each sensor node only communicates with its closest neighbors, and only one sensor node at a time sends aggregated data to the base station. However, LEACH and PEGASIS do not fully consider the energy hole problem. LEACH rotates the cluster head, whereas PEGASIS changes its designated node. We can further improve the energy efficiency of LEACH and PEGASIS, because they are not optimized to minimize the total communication energy consumption of the WSN. Other studies focused on minimizing the total communication energy consumption using graph-theoretic structures to form a minimum spanning tree. Solutions include using the Prim, Kruskal and Dijkstra routing algorithms [10]. These algorithms can construct tree topologies that contain all the nodes and minimize the sum of the edge lengths (communication distances). Although these algorithms effectively minimize the energy consumed by data communications, they do not solve the energy hole problems, so a

premature shutdown of the network is possible.

Another group of studies concentrated on solving the energy hole problem, including those by Olariu and Stojmenovic [4], Wu et al. [11], and Lian et al. [12]. Olariu and Stojmenovic identified the energy hole problem. They showed that if the energy consumption grows proportionally faster than the second power of the communication distance, we can prevent non-uniform energy consumption using an appropriate design. For example, we can place data sinks at strategic locations and adjust the network radius around the sinks. Although the energy expenditure in such a system is balanced across the network, it is suboptimal. Wu et al. proposed regulating the number of nodes in each corona, and setting the ratio of node densities in two adjacent coronas to mitigate the energy hole problem. Lian et al. proposed moving the sinks to balance the load across the deployment area. LEACH [8] is another attempt at rotating cluster heads and clusters to avoid non-uniform energy consumption, because a cluster head consumes more energy. Each node independently decides whether to be a cluster head. The cluster heads and their corresponding clusters are rotated in a random manner. Unfortunately, these approaches are not adequately energy efficient.

Some researchers focused on enhancing the fault tolerance of data communications. There are two methods for handling the fault tolerance problem [3]: (1) proactive approaches that attempt to provision resources in the network topology to tolerate node failures, and (2) reactive approaches that attempt to tolerate node failures through real time restoration of lost connectivity. In this paper, we focused on the proactive approach. There are two variants to the proactive approach. In the first, fault tolerant topologies are formed at the time of deployment. We did not consider this approach because WSN nodes in infrastructures such as bridges should be deployed at fixed positions. In the second, we form a k-vertex connected (referred to as k-connectivity) WSN, or designate backup nodes for critical nodes in the network. A k-connectivity network has a $k-1$ fault tolerance, which means it can tolerate the failure of up to $k-1$ nodes [13].

An algorithm aimed at reducing communication energy while preserving network connectivity was proposed by Li and Hou [14]. They presented a centralized algorithm called the fault-tolerant global spanning subgraph ($FGSS_k$), which was based on Kruskal's algorithm. Different components are iteratively merged until only one k-connected component remains. A localized algorithm called the fault-tolerant local spanning subgraph ($FLSS_k$) was based on $FGSS_k$, and proposed to control the topology. The authors showed that $FGSS_k$ and $FLSS_k$ can preserve k-connectivity and minimize the maximum transmission power of the network. Moreover, $FLSS_k$ maintains bi-directionality for all links in the topology. Li et al. investigated the minimum transmission range that guarantees k-connectivity with a high probability [15]. They also presented a localized method for controlling the network topology that was based on the Yao structure and preserves k-connectivity. Jorgic et al. proposed an alternative fault tolerance method that uses localized algorithms to detect the critical nodes and links [16]. Other works have attempted to generate k-connectivity, including [17, 18, 19].

Other proactive approaches designate spares for critical nodes, which act as cut vertexes in the network topology. The simplest way to achieve this is to place redundant nodes that accompany the critical nodes. However, if all the nodes serve the application, the spare nodes must be selected from the active nodes. In [20], Vaidya and Younis proposed NORAS to select spare nodes within the 2-hop neighborhood of a failed critical node. If non exist, the search widens to include more distant nodes. When a critical node fails and is detected, the designated spare nodes travel to replace the critical node, or a series of cascaded relocations on the shortest route between the critical and selected spare nodes are triggered to split the travelling distance. In [21], the authors proposed PADRA, which identifies a connected dominating set for the WSN. PADRA designates a failure handler to each critical node in case it fails. An ideal handler will be a dominatee neighbor of the critical node that can simply replace the critical node. In our approach, we balanced the energy efficiency, energy hole problem, and fault tolerance to obtain reliable energy efficient wireless communications.

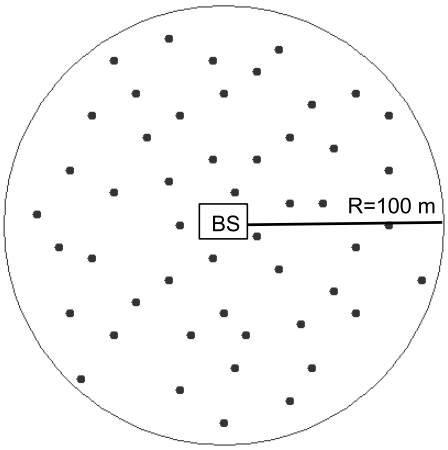

Figure 1: Wireless sensor network configuration.

3 Energy Efficient Routing Algorithm

3.1 Preliminaries

We considered a circular WSN with a 100-m radius, a random number of uniformly and randomly distributed homogeneous sensor nodes, and a base station at the center, as shown in Fig. 1. Compared with the WSN used in the LEACH and PEGASIS studies where the sensor nodes were deployed in a square space and were at least 100 m from the base station, our WSN configuration is closer to actual WSN systems such as those using ZigBee[1] devices, which have a maximum communication distance of 100 m. We also assumed that the base station is powered by a stable power source, so the energy consumed by the base station is separate from that consumed by the wireless communications. We used a free space radio propagation model to calculate the transmitting and receiving costs for a k-bit message at a distance of d. That is,

Transmitting

$$E_{T_x}(k,d) = E_{elec} * k + \epsilon_{amp} * k * d^2, \quad (1)$$

Receiving

$$E_{R_x}(k) = E_{elec} * k. \quad (2)$$

These are the same as those used in the LEACH and PEGASIS studies, where $E_{elec} = 50\ nJ/bit$ for running the transmitter or receiver circuitry and $\epsilon_{amp} = 100\ pJ/bit/m^2$ for running the transmit amplifier. Because we only focused on extending the WSN lifetime through the use of effective routing algorithms, we ignored other energy consumptions such as those of the CPU and sensors. Additionally, for simplicity, we assumed that each node must send 2000 bits of data as in the LEACH and PEGASIS studies, and that this amount stays the same despite data aggregations.

3.2 Wireless Communication Graph

The initial wireless sensor network has N nodes and can be described as an undirected weighted graph, $G =< V, E >$, where $V = \{0, ..., N-1\}$ is a vertex set indicated by the sensor nodes, and E is an edge set showing the data communication between two nodes. An edge in the graph is indicated by $(i, j) \in V$. Because the data sent between two nodes determines the direction of an edge, the wireless sensor network graph is a directed graph. We let $dir(i, j)$ denote that the direction of edge (i, j) is from i to j. To calculate the energy consumed by the data communication between nodes i and j, we use the energy consumption metrics in Equations (1) and (2) to define a nonnegative weight, $w(i, j) = E_{T_{(i,j)}} + E_{R_{(i,j)}}$, which is associated with each edge $(i, j) \in V$ with $dir(i, j)$. Note

[1]ZigBee is a registered trademark of the ZigBee Alliance.

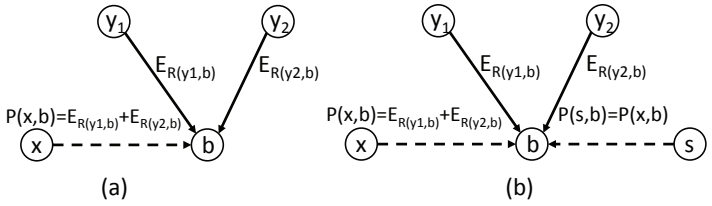

Figure 2: Graphical representation of phony weight.

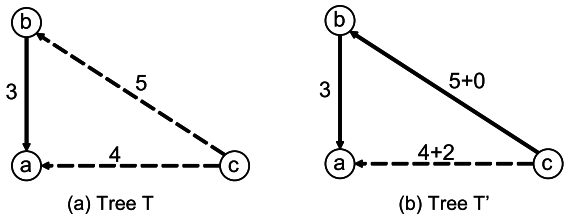

Figure 3: Phony weight.

that the notation $E_{T_{(i,j)}}$ is the transmitting energy of node i and $E_{R_{(i,j)}}$ is the receiving energy of node j. The total energy consumed by the wireless communication between i and j is $w(i,j)$. The value of $w(i,j)$ is determined if the network topology is determined, because it only depends on the distance between i and j.

3.3 Energy Hole Aware and Energy Efficient Communication Routing Algorithm

Our proposed EHAEC routing algorithm solves the energy hole problem without changing the network configuration as much as possible, while minimizing the total energy consumption of the wireless communications. Note that EHAEC achieves a trade-off between solving the energy hole problem and optimizing the total energy consumption of the wireless communications. The total energy consumption cannot be minimized if the energy hole problem is considered, because avoiding an energy hole requires the communication between nodes that increases the energy consumption. However, EHAEC can minimize the energy consumed by sending and relaying data.

EHAEC is based on Prim's algorithm [10], which finds the optimal "minimum spanning tree" for an undirected connected weighted graph. EHAEC is different from Prim's algorithm in that it uses a "phony weight" to avoid energy holes as much as possible. Suppose that edges (y_i, b) with $dir(y_i, b)$ exist in a "minimum spanning tree" T, $y_i \in T$, and $i \in N$. A phony weight for $b \in T$ and $x \in V \backslash T$ is defined as

$$P(x,b) = \sum_{y_i \in T} E_{R_{(y_i,b)}}. \qquad (3)$$

The direction of edge (x,b) is $dir(x,b)$. Phony weight $P(x,b)$ is added to $w(x,b)$, where edge (x,b) may connect an unconnected node x to a node b in the tree that has already relayed data from other nodes. The use of phony weights prevents the situation in which one node relays data from many nodes, to uniformly distribute the energy consumed by sending and relaying data. However, phony weight $P(x,b)$ is not actually consumed by edge (x,b), because it has already been consumed by edges (y_i,b). Figure 2(a) shows a graphical representation of Equation (3), where $i = 2$. It shows that, $P(x,b) = E_{R_{(y_1,b)}} + E_{R_{(y_2,b)}}$ when node x tries to connect to node b, and nodes y_1 and y_2 are already in the tree. This means that the value of the phony weight depends on the number of nodes connected to node b at time t. Therefore, $P(x,b) = P(s,b)$ for any node $s \in V \backslash T$ at time t when only nodes y_1 and y_2 are connected to node b (Fig. 2(b)). Let us consider an example of the effect of using a phony weight. Figure 3 shows the procedure for forming T' from T by adding node c, where

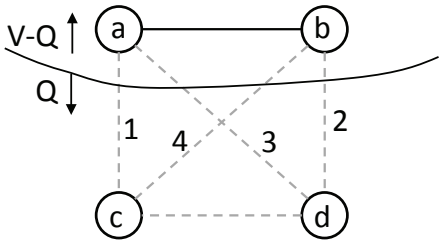

Figure 4: Explanation of EXTRACT-MIN(Q).

node a is the data sink. In Fig. 3(a), nodes a and b were added to tree T. If we next try to add node c to T, we should either add it to edge (c, a) or (c, b). According to the definition of a phony weight, because node a has already relayed data from node b, the other edges that may relay data to node a should add the phony weight to themselves. In Fig. 3(b), assuming the phony weight is 2, we have $P(c, a) = E_{R_{(b,a)}} = 2$ and $P(c, b) = 0$. Therefore, $w(c, a) = 6$ and $w(c, b) = 5$. Because $w(c, b) < w(c, a)$, edge (c, b) is added to tree T'.

Algorithm 1: EHAEC(G,w,r)

```
01  Q ← V(G)
02  for each u ∈ Q
03      key[u] ← ∞
04      π[u] ← NIL
05  key[r] ← 0
06  while Q ≠ 0
07      u ← EXTRACT-MIN(Q)
08      A ← {(u, π[u]) : v ∈ V − r − Q}
09      edge direction: dir(u, π[u])
10      for each v ∈ Adj[u]
11          for each s ∈ Adj[π[u]] − u and s ∈ V\A, s ≠ r, u ≠ r, π[u] ≠ r
12              P(s, π[u]) = Σ_{x∈A} E_R(x, π[x]), where π[x] = π[u]
13              w(s, π[u]) = w(s, π[u]) + P(s, π[u])
14              if s ∈ Q and π[s] = π[u] and key[s] is w(s, π[u]) that in last round
15                  key[s] ← w(s, π[u])
16          if v ∈ Q and w(v, u) < key[v]
17              π[v] ← u
18              key[v] ← w(v, u)
19  return A
```

EHAEC works in the following way. The required inputs are graph G of the WSN, and root r (the base station) of G. The output returns spanning tree A, which avoids energy holes as much as possible without reconfiguring the network. In EHAEC, all vertexes not in the tree are stored in a priority queue, Q. The weight of each edge are calculated a prior. Weight of an edge (v, u) is defined as $w(v, u) = E_{T_{(v,u)}} + E_{R_{(v,u)}}$. For each vertex v, $key[v]$ is the minimum weight of any edge connecting v to a vertex in the tree. Field $\pi[v]$ indicates the parent of v in the tree. Lines 1-5 of the algorithm are the initializations. While Q is not empty, Line 7 extracts a vertex $u \in Q$ incident on a light edge crossing the cut $(V - Q, Q)$ (with the exception of the first iteration, in which $u = r$ due to line 5)[2], which means it extracts a node u which should be add to the tree in the current loop. Line 8 stores the parent-child vertex set in tree A. Line 09 defines the direction of

[2]We use Fig. 4 to explain EXTRACT-MIN(Q). Supposing vertexes a and b are in the tree. Hence, $Q = \{c, d\}$ and $V - Q = \{a, b\}$. The black curve indicates the idea of cut $(V - Q, Q)$. Since EXTRACT-MIN(Q) extracts a light edge crossing the cut, vertex c is extracted because edge (a, c) is the light edge among edges (a, c), (a, d), (b, c) and (b, d).

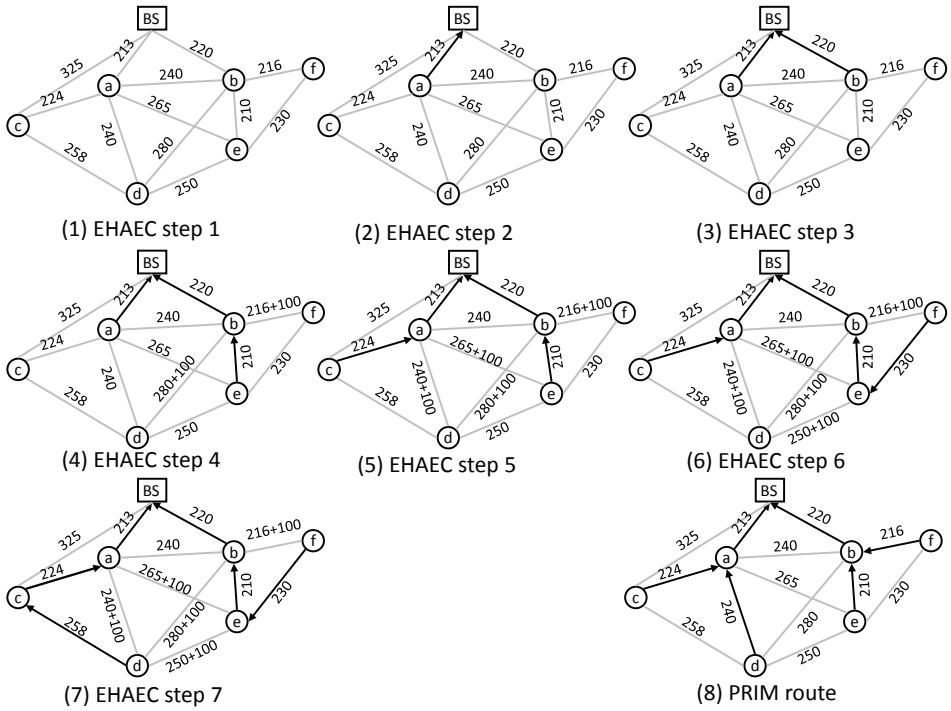

Figure 5: EHAEC and Prim Routing.

edge $(u, \pi[u])$. For every vertex v that is adjacent to u but not in the tree (line 10), line 12 defines the phony weight $P(s, \pi[u])$ and line 13 updates the edge weight $w(s, \pi[u])$ if vertex s is adjacent to $Adj[\pi[u]] - u$, and vertexes s, u and $\pi[u]$ are not the root (line 11). Except for root r, if a parent $\pi[u]$ is already connected to a child x, phony weight $P(s, \pi[u])$ is defined to avoid connecting another vertex s to $\pi[u]$ because this may cause an energy hole in vertex $\pi[u]$. Lines 14 and 15 determine if $key[s]$ should be updated. In Lines 16-18, for every vertex v that is adjacent to u but not in the tree (line 10), if edge (v, u) has the smallest weight, the key and π fields of every vertex v are updated. The updated key contains the vertex which has the minimum weight edge that may be extract in the next loop. Finally, in Line 19, EHAEC returns the spanning tree A. EHAEC has the same complexity as Prim's algorithm, which is $O(|E| \lg |V|)$.

Examples of applying the EHAEC are shown in Figs. 5(1-7). Figure 5(1) shows a unit weighted graph $G =< V, E >$, where $V = \{BS, a, b, c, d, e, f\}$, vertex BS is the root of the tree, and the other vertexes are sensor nodes that must be connected to the tree. Note that there are other edges in graph G, for example, (BS, f), (b, c), and (d, f). However, these edges have large weights, so we have ignored them. The weight is shown for each edge, for example, $w(a, b) = 240$ means that the weight of edge (a, b) is $240\mu J$. These values are calculated using Equations (1) and (2). We can now show how EHAEC forms a route. In the first step shown in Fig. 5(1), $u = r = BS$ because of Lines 5 and 7, hence the starting vertex of the tree forming procedure is BS. Therefore, BS is on the tree. Since $u = r = BS$ in line 11, lines 12 to 15 are not executed. For every vertex $v \in Adj[BS] = \{a, b, c\}$, line 16 judges whether $w(a, BS)$, $w(b, BS)$ and $w(c, BS)$ are less than $key[a]$, $key[b]$ and $key[c]$, respectively. Line 16 is true since the key fields are infinities, hence, $\pi[a] = BS$, $\pi[b] = BS$ and $\pi[c] = BS$ in line 17; $key[a] = w(a, BS) = 213$, $key[b] = w(b, BS) = 220$ and $key[c] = w(c, BS) = 325$ in line 18. In step 2 shown in Fig. 5(2), $u = a$ in line 7 since $key[a] = w(a, BS) = 213$ is minimum in step 1. Line 8 stores edge (a, BS) to tree A and line 9 decides the direction of edge (a, BS) is $dir(a, BS)$ since $\pi[a] = BS$ in step 1. Since $\pi[u] = r = BS$ in line 11, lines 12 to 15 are not executed. line 16 judges whether $w(b, a)$, $w(c, a)$ and $w(d, a)$ are less than $key[b]$, $key[c]$ and $key[d]$, respectively. Line 16 is only true for vertexes c and d. Hence, $\pi[b] = BS$ which is not change, $\pi[c] = a$ and $\pi[d] = a$ in line 17; $key[b] = w(b, BS) = 213$ which is not

change, $key[c] = w(c, a) = 224$ and $key[d] = w(d, a) = 240$ in line 18. The same procedure repeats until step 3 shown in Fig. 5(3). In step 4 shown in Fig. 5(4), after $u = e$ is extracted in line 7, $s \in Adj[\pi[u]] - u = Adj[\pi[e]] - e = Adj[b] - e = \{d, f\}$ in line 11. Therefore, $P(d, b) = E_R(e, b) = 100$ and $P(f, b) = E_R(e, b) = 100$ in line 12. $w(d, b)$ and $w(f, b)$ are updated in line 13. Line 14 judges whether $key[d]$ is $w(d, a)$, and $key[f]$ is $w(f, b)$, where $w(d, a)$ and $w(f, b)$ are values in the previous step. Since it is true that $key[f]$ is $w(f, b)$ which in the previous step, $key[f] = w(f, b)$ in line 15, the $w(f, b)$ here is the value calculated in line 13, hence the key field is updated. The rest part of step 4 is similar with that in step 1 and 2. The same procedure repeats in step 5, 6 and 7. In step 7 shown in Fig. 5(7), since all the nodes are connected by the EHAEC, a spanning tree has been formed. Compared with the route generated by Prim's algorithm in Fig. 5(8), EHAEC avoided the energy holes in nodes a and b. Moreover, EHAEC was previously proved to construct a communication route that consumes the least energy in sending and relaying data in the WSN [1], because the advantage of using phony weights is that the internal nodes do not consume more energy when relaying data in every node-adding procedure. Please refer to [1] for more detailed information about theorems related to EHAEC.

4 EHAEC for One-Fault Tolerance

Our proposed algorithm for one-fault tolerance (EHAEC-1FT) maintains the two-connectivity of the network, because it can tolerate the failure of one node in the WSN. When the requirement is more than two-connectivity, it can be extended to X-nFT, which can be used by any spanning tree routing algorithm X for n-fault tolerance.

Algorithm 2: EHAEC-1FT(A,w,r)

```
01  for each u ∈ A
02      key[u] ← ∞
03  for each y ∈ A\{x}, where x ∈ A and π[y] = x
04      for each z ∈ Adj[x] − y
05          for each s ∈ Adj[π[y]] − y and s ∈ A\1FTSG, s ≠ r, y ≠ r, π[y] ≠ r
06              P(s, π[y]) = Σ_{a∈1FTT} E_R(a, π[a]), where π[a] = π[y]
07              w(s, π[y]) = w(s, π[y]) + P(s, π[y])
08              if s ∈ Q and π[s] = π[y] and key[s] is w(s, π[y]) that in last round
09                  key[s] ← w(s, π[y])
10          if z ∈ A\{x, y} and w(z, x) < key[z]
11              π[z] ← x
12              key[z] ← w(z, x)
13      1FTSG ← {(z, π[z]) : z ∈ A\{r, x, y}}
14      edge direction: dir(z, π[z])
15  return 1FTT = A ∪ 1FTSG
```

EHAEC-1FT uses EHAEC to find a subgraph of G that, in union with the graph of set A obtained by EHAEC, can maintain two-connectivity and one-fault tolerance. We call this subgraph a "one-fault tolerant subgraph (1FTSG)" and the union graph a "one-fault tolerant tree (1FTT)". The input to EHAEC-1FT is graph A of EHAEC, and the output is the union graph 1FTT. The basic idea of EHAEC-1FT is to first suppose that every node in A has failed, and then find new parent nodes for the child nodes that belonged to the failed nodes. Note that if a node fails, all the edges were connected to the failed node are deleted from the tree. In EHAEC-1FT, Lines 1 and 2 initiate the edge weights. Line 3 identifies the child nodes of the failed nodes, and Lines 4-14 use EHAEC to find new parent nodes for only these child nodes. Note that Lines 4-12 and 13-14 in EHAEC-1FT have the same function as Lines 10-18 and 8-9 in EHAEC, only the parameters are different. Finally, Line 15 returns tree 1FTT. Because Lines 3-4 are executed $O(|V|)$ times, and Lines 5-15 are executed $O(|E| \lg |V|)$ times, the complexity of EHAEC-1FT is $O(|V||E| \lg |V|)$.

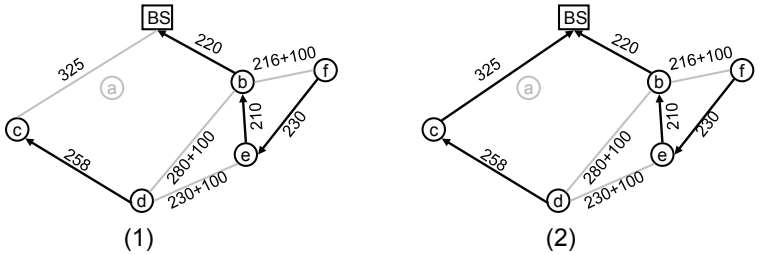

Figure 6: EHAEC-1FT example.

Table 1: 1FTSG lookup table

Failed node	Affected nodes	Added edge
a	c,d	(c, BS)
b	e,f	(e, d)
c	d	(d, a)
d	NIL	NIL
e	f	(f, b)
f	NIL	NIL

The EHAEC-1FT example in Fig. 6 uses the output A of EHAEC that is shown in Fig. 5(7). Figure 6(1) shows the graph when node a has failed. The edges connected to a are deleted because of this failure. The failure of node a means that c and d are removed from the communication tree. Therefore, node c takes BS as its new parent node and maintains the connectivity of the communication tree (Lines 4-14). Therefore, edge (c, BS) with $dir(c, BS)$ is added to subgraph 1-FTSG in Fig. 6(2). Similarly, if nodes $b, c, d, e, and\ f$ failed in turn, new edges would be added to 1-FTSG, as shown in Table 1. Consequently, 1FTSG is constructed as in Fig. 7(1). Moreover, the combination of 1FTSG with the tree formed by EHAEC constructs a 1FTT (Line 15), as shown in Fig. 7(2).

As mentioned above, EHAEC-1FT can be extended to X-nFT to maintain n-fault tolerance for any tree routing algorithm X. EHAEC-1FT can be extended to satisfy the requirement of tolerating an n-node failure, as shown in Algorithm 3. X-nFT finds all the child nodes of n failed nodes (Lines 1 and 2), and then uses a particular tree routing algorithm (Line 3) to build new connections between these child nodes and other nodes in the tree, resulting in n-fault tolerance. For every $i = 1...n$, X-nFT returns an i-fault tolerant subgraph, $iFTSG$ (Line 4). Finally, X-nFT returns the union graph $nFTT$, which is n-fault tolerant.

Different systems have different fault tolerant redundancy requirements. There are two ways to make a system fault tolerant: create a fault tolerant redundancy when or before a failure (standby

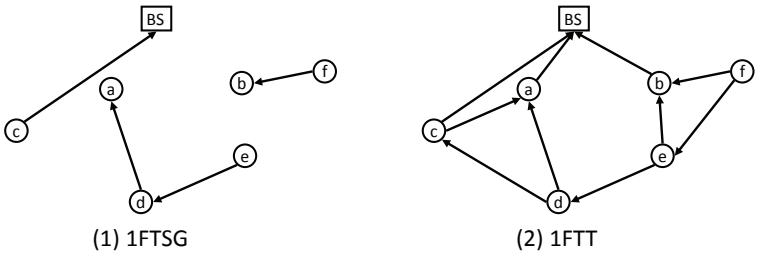

Figure 7: Graph of 1FTSG and 1FTT.

Algorithm 3: X-nFT(A,r)

```
01 for i = 1 to n
02     for all y_i ∈ A\{x_i}, where x_i ∈ A and π[y_i] = x_i
03         do TreeRoutingAlgorithm(A,r)
04             return iFTSG
05 return nFTT = A ∪ 1FTSG ∪ 2FTSG ∪ ... ∪ nFTSG
```

fault tolerance, STB), or constantly have a fault tolerant redundancy (active fault tolerance, ACT). With STB, the WSN uses tree A of EHAEC for data communication. If n nodes fail, only the parts of subgraphs 1FTSG...nFTSG that can tolerate the failed nodes are used with tree A. ACT uses tree nFTT, which is a complete combination of tree A and the fault-tolerant subgraphs.

5 Active Spare Selecting Algorithm

We considered another provisioned tolerance scheme that finds spare nodes for the failed node, which serve as cut vertexes. In this situation, the spare node takes the responsibility of the failed node. In most related works, spare nodes are additional nodes that placed at right positions accompany with the failed nodes. However, in WSN systems for infrastructures such as bridges and tunnels, it is difficult to place additional nodes because we do not know which node is likely to fail. Placing spares around some critical node is risky. Moreover, additional spare nodes increase costs. Therefore, in this section, we present the ASSA, which attempts to select appropriate nodes in the network to be used as active spare nodes. The spare nodes only substitute the failed critical nodes. Here, critical nodes are nodes that act as cut-vertexes in the network, i.e., the internal nodes. When a critical node fails, the ASSA selects an active spare node to take on its responsibilities.

Algorithm 4: ASSA(A,α,β)

```
01 input impact factor α and β, where α + β = 1
02 for each u ∈ A\L
03     CL(u) = NumOfSuccessor(u)
04 for each y ∈ A\L, where π[y] = z
05     for each x ∈ Adj[y]
06         CL_TEMP[x] = CL(x) + CL(y) + 1
07         w(x, y) = E_{T(x,y)} + E_{R(x,y)}
08     calculate NCL = 1/RMS[CL_TEMP] and Nw = 1/RMS[w]
09     for each x ∈ Adj[y]
10         NEC(x, y) = α * NCL(x) + β * Nw(x, y)
11 for each z ∈ A\L, where π[y] = z, and π[z] ≠ x
12     if z failed
13         for each y ∈ A\L, where π[y] = z, start the loop with CL(y) in decreasing order
14             if ∃x ∈ A\L, where NEC(x, y) is minimum
15                 π[y] = x
16                 CL(x) = CL_TEMP(x)
17                 PRCT ← {(y, π[y])}
18 return PRCT
```

In the ASSA, we use the concept of a sensor node's critical level $CL(x)$, which is the number of successor nodes to x ($CL(x) = NumOfSuccessor(x)$). Generally, a node is more critical if it transmits data from more child or successor nodes. Therefore, it has a higher critical level if it has more successors. Moreover, because the ASSA is designed to select active spares using a combined weight of the node's critical level and edge weight, we also use an evaluation criterion $EC(x, y) = \alpha * CL(x) + \beta * w(x, y)$. Suppose that node z fails and y is a child node of z. For a spare

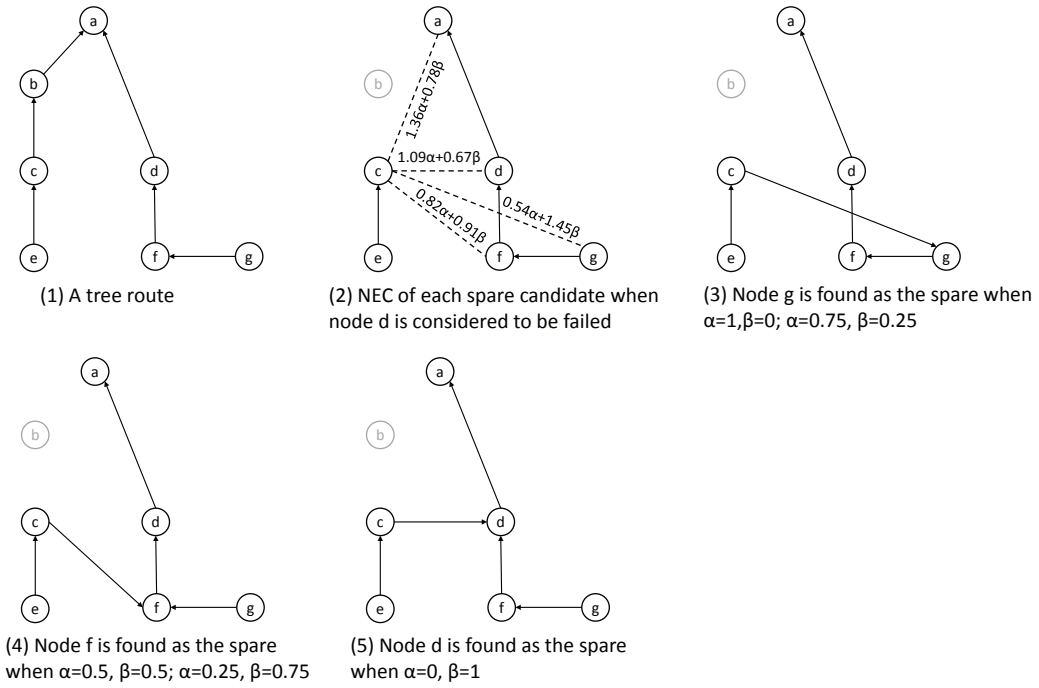

Figure 8: ASSA example.

node candidate x, the evaluate criterion considers the weight of the critical level of node x and the edge weight $w(x, y)$. Because $CL(x)$ and $w(x, y)$ are different metrics, we normalize the values using $N = \frac{1}{RMS[x]}$ so that they are comparable. $RMS[x] = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2}$ indicates the root mean square, which is a common normalization method. Hence, $CL(x)$ and $w(x, y)$ are normalized to $NCL(x)$ and $Nw(x, y)$, respectively, and the normalized $EC(x, y)$ is $NEC(x, y) = \alpha * NCL(x) + \beta * Nw(x, y)$. Moreover, we set $\alpha + \beta = 1$. System designers can set the values of α and β to express their inclination towards fault tolerance or energy efficiency.

We can now explain how the ASSA works. First, in Line 1, the values of α and β are chosen by the system designer. Line 3 initializes the critical levels of all nodes u in tree A (except the leaf node set L). Lines 4 to 7 define a temporary critical level value ($CL\_TEMP[x]$) for each node x that is a neighbor to node y such that y's current parent node is z. We set $CL\_TEMP[x]$ in this way so that, if node y's parent node z fails and node y takes node x as its new parent, then the critical level of x is updated. Line 7 sets the edge weight $w(x, y)$, which indicates the energy consumption for data communication between nodes x and y. Note that $w(x, y)$ does not include the phony weight, because the ASSA uses the critical level. Line 8 calculates the normalized CL_TEMP and $w(x, y)$. Lines 9 and 10 calculate the normalized evaluation criterion, $NEC(x, y)$, which determines if x should be selected as be the active spare node of z. In Lines 11 to 17, if any node z has failed, all its child nodes (y) will find the new proper parent node x that has the minimum $NEC(x, y)$. This procedure starts with node y in descending order of $CL(y)$. Line 15 sets x as the new parent node of y, and Line 16 updates $CL(x)$. Line 17 stores the new link relation of $\{(y, \pi[y])\}$ to $PRCT$, where $PRCT$ is defined as the partition route changed tree. Finally, Line 18 returns $PRCT$. We calculated the complexity of ASSA in three parts. First, Lines 2 and 3 are executed $O(|A - L|)$ times. Next, for Lines 4-10, Line 4 is executed $O(|A-L|)$ times, and Lines 5-7 and 9-10 are executed $O(|E|)$ times. Therefore, the second part is executed $O(|E||A-L|)$ times. In the third part, Line 11 is executed $O(|A - L|)$ times, and Lines 13-17 are executed $O(|A - L|)$ times. Therefore, the third part is executed $O(|A - L|^2)$ times. Overall, the complexity of ASSA is $O(|A - L|^2)$.

We now give an example that uses ASSA, as in shown in Fig. 8. Figure 8(1) shows a tree route generated by a spanning tree algorithm. In Fig. 8(2), an internal node b has failed. Because

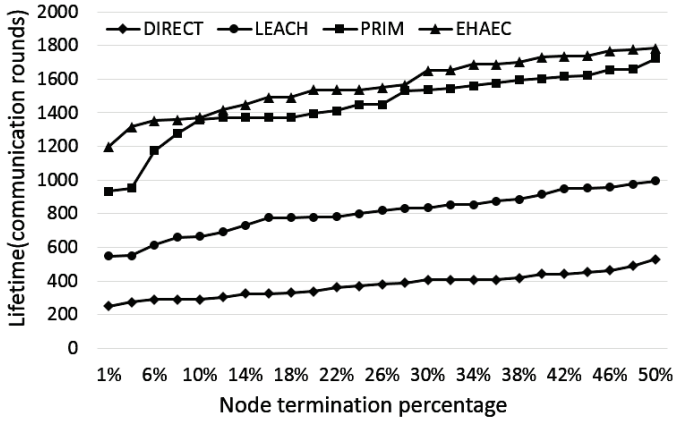

Figure 9: Simulated lifetime for the first situation (energy efficiency is more important than fault tolerance).

Table 2: Efficiency when energy efficiency is more important than fault tolerance.

Shutdown percentage	Direct	LEACH	Prim	EHAEC
1%	1	2.2	3.7	4.8
25%	1	2.2	3.8	4.1
50%	1	1.9	3.3	3.4

active spares should be selected so that node c can tolerate the failure of node b, the normalized evaluate criterion $NEC(x, y)$ for each possible link between the child nodes and their neighbors are calculated in Line 10. By comparing the values of $NEC(x, y)$ for different α and β, the appropriate spare nodes are found using lines 14-17. Figures 8(3-5) show the new routes built by adapting active spares when $\alpha = 1$ and $\beta = 0$, $\alpha = 0.75$ and $\beta = 0.25$, $\alpha = 0.5$ and $\beta = 0.5$, $\alpha = 0.25$ and $\beta = 0.75$, and $\alpha = 0$ and $\beta = 1$, respectively.

The ASSA considers both fault tolerance and energy efficiency, and can be adapted to different applications. First, we must consider the situations when either fault tolerance or energy efficiency is more important. Energy efficiency is considered more important when the WSN environment is stable. For instance, when there are no or very few signal interferences, sensor nodes are rarely damaged, or there are short communication distances. Fault tolerance is considered more important in the opposite case, when the environment is not stable, or when sensor nodes are not independent of each other. For instance, if an internal node only can send its data when it has already received its child nodes' data. Secondly, we also considered some concrete situations of ASSA, for example, when the number of nodes is large, or a high density of nodes in a WSN. If the number of nodes is large, it is easier to find child nodes, and $NCL(x)$ is relatively large. On the contrary, the communication distance is short if the number of nodes is large or the WSN is dense, so $Nw(x, y)$ is small. Therefore, $NCL(x)$ is the dominant factor that affects the value of $NEC(x, y)$ if $\alpha \neq 0$, in this case. Hence, if the energy efficiency is more important than fault tolerance, $\alpha = 0$ and $\beta = 1$ are effective. Alternatively, we may have a low density or small WSN. In this case, $NCL(x)$ is small and $Nw(x, y)$ is large. Consequently, we can set $\alpha = 1$ and $\beta = 0$ if a WSN requires fault tolerance more than energy efficiency.

6 Simulation

We evaluated the effectiveness of our algorithms by simulating the lifetime of an entire WSN, and compared it with the lifetime of each node when using direct data transmission and Prim's algo-

Table 3: Efficiency when fault tolerance is more important than energy efficiency.

Stoppage percentage	EHAEC	EHAEC-1FT-STB	EHAEC-1FT-ACT	ASSA	$FGSS_2$
1%	1	1	0.41	1	0.38
25%	1	1.17	0.53	1.25	0.51
50%	1	1.23	0.55	1.32	0.37

rithm. In direct data transmission, sensing data are directly transmitted from each node to the base station. Because we are concerned with energy efficiency and fault tolerance, we used two situations when evaluating EHAEC, EHAEC-1FT, and ASSA. That is, we considered when: 1) energy efficiency is more important than fault tolerance, and 2) fault tolerance is more important than energy efficiency. We simulated a 50-node WSN configured as shown in Fig. 1, for both situations. The lifetime of a battery-powered WSN is defined as the number of communication rounds till λ% of the sensor nodes stop operating, where λ is specified by the system designer [22]. We assumed that the WSN remained functional as long 50% of the nodes were operating. We also assumed that each node had an energy capacity of $0.5J$. We calculated the lifetime of the nodes by first calculating the energy consumption of each node for different routes using Equations (1) and (2), and then calculated the number of data communication rounds (lifetime) for each node using $(node\ energy\ capacity)/(node\ energy\ consumption)$.

6.1 Energy-Efficiency-First Evaluation

In the first situation (energy efficiency is more important than fault tolerance), if an internal node stops operating, its child nodes must find another node to relay their data. In this simulation, the communication route is rebuilt by the routing algorithms if a node stops operating, and the energy consumption of each node changes in accordance with the rebuilt route. The algorithm uses this route rebuilding process to ensure that we use the entire the energy capacity of each sensor node even if its parent node has stopped operating. Figure 9 shows the simulation results for communication using direct data transmission and using routes generated by LEACH [3], Prim's algorithm, and EHAEC. Table 2 shows the corresponding efficiencies, where efficiency is defined as the WSN lifetime ratio of $(routing\ algorithm)/(Direct)$, i.e., the efficiency of direct data transmission is 1. These results show that EHAEC, Prim's algorithm, and LEACH performed much better than direct data transmission, and that EHAEC outperformed Prim's algorithm and LEACH. Moreover, the gradients of the EHAEC and Prim curves show that EHAEC can balance the lifetime of each node (i.e., avoid energy holes) better than Prim's algorithm. The gradients of the EHAEC and LEACH were very similar because LEACH also attempts to avoid energy holes by changing clusters and cluster heads. However, the WSN lifetime was much longer using EHAEC than LEACH. Our simulation results showed that EHAEC performed twice as well as LEACH.

6.2 Fault-Tolerance-First Evaluation

For the second situation (fault tolerance is more important than energy efficiency), we assumed that a node had stopped operating when there were no possible edges for it to transmit its data to the base station, even if the node had not run out of energy. Consider the example in Fig. 7(2). Although the route guarantees 2-connectivity, if nodes b and d have stopped, nodes e and f are also considered to have stopped because they cannot transmit data to the base station. For the route generated by EHAEC, we compared the ASSA and EHAEC-1FT with the standby and active fault tolerances using EHAEC (which does not consider fault tolerance). We used $\alpha = 0$ and $\beta = 1$ in this simulation, considering that we wanted the best performance in terms of energy efficiency. We also compared these results with $FGSS_k$. Although $FGSS_k$ can find a k-connected

[3] Although our algorithms and LEACH use different network topologies, in these simulations we applied LEACH's idea to our WSN topology, as shown in Fig. 1.

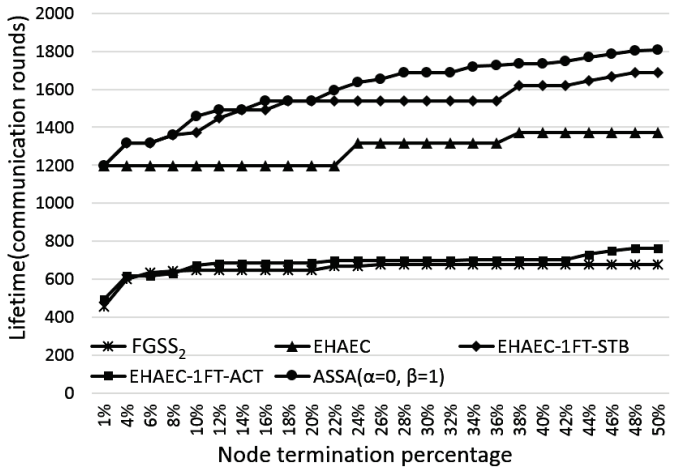

Figure 10: Simulated lifetime for the second situation (fault tolerance is more important than energy efficiency).

spanning subgraph, we assumed that it maintained 2-connectivity in the simulations so that we could compare it with our algorithms. Therefore, $FGSS_k$ is $FGSS_2$ in these simulations. Figure 10 shows the simulation results. Table 3 shows the corresponding energy efficiencies. With standby fault tolerance, EHAEC-1FT was more efficient than EHAEC, and with active fault tolerance it was less efficient. This is because the communication tree used with the active fault tolerance was larger than that used with the standby fault tolerance. Therefore, it consumed more energy. From the fault tolerance perspective, EHAEC cannot tolerate node failure whereas EHAEC-1FT can tolerate one node failure using standby and active fault tolerance. Moreover, the ASSA was more efficient than the other three methods, because the batteries of all the nodes were used and because it can tolerate all node failures at any time. EHAEC-1FT and ASSA are more efficient than $FGSS_2$. However, the results of $FGSS_2$ and EHAEC-1FT in active mode are almost the same. This is because $FGSS_2$ generates a 2-connected spanning tree based on Kruskal's algorithm, and the redundant routes are always applied during wireless communications. Next, we investigated the fault tolerant effects of the algorithms by comparing the WSN termination speeds in Fig. 11. The speed of the WSN termination is defined as $speed = \frac{lifetime_{longest}}{lifetime_{self}}$, where $lifetime_{longest}$ is the longest lifetime of the five simulated algorithms under the current WSN lifetime definition (i.e., the length of time until $\lambda\%$ of sensor nodes stop operating), and $lifetime_{self}$ is the lifetime of a WSN using one of the algorithms. A quicker speed corresponds to a worse fault tolerance. Therefore, Fig. 11 shows that the ASSA performed the best and $FGSS_2$ was the worst.

We also compared the energy efficiency and failure times using different values of α and β: $\alpha = 1$ and $\beta = 0$, $\alpha = 0.5$ and $\beta = 0.5$, and $\alpha = 0$ and $\beta = 1$. Figures 12 and 13 show the results. In Fig. 12, there were no significant differences when using different values of α and β (because the ASSA finds spare nodes for the route generated by EHAEC, which is originally energy efficient). However, the ASSA results appears to be more energy efficient for larger β. There were less node failures when α was larger (Fig. 13). The result agrees with idea behind the ASSA.

7 Conclusion

We investigated the energy efficiency and fault tolerance of WSNs, given that the communication distances and energy hole problem are critical to the battery life of the sensor nodes, which affects the network's lifetime. Our EHAEC routing algorithm was based on Prim's algorithm and achieves energy efficient wireless communications by minimizing the energy consumed by sending and relaying data. It avoids energy holes to the maximum extent possible without reconfiguring the network,

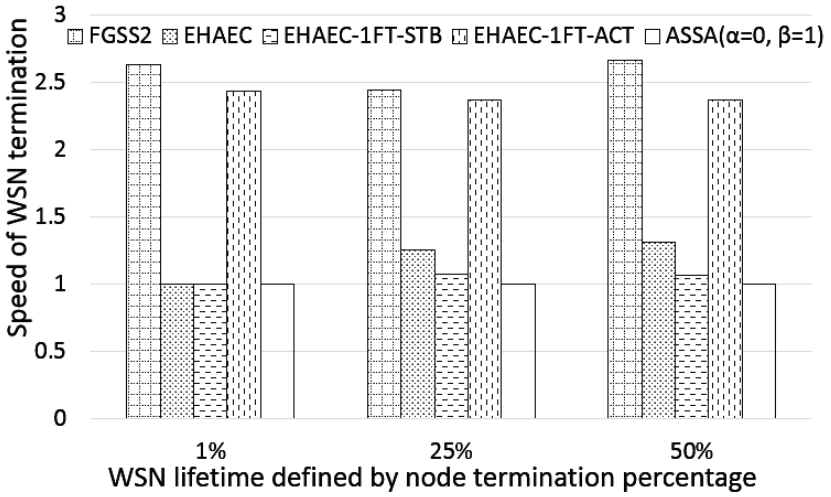

Figure 11: Simulated fault tolerant effect when fault tolerance is more important than energy efficiency.

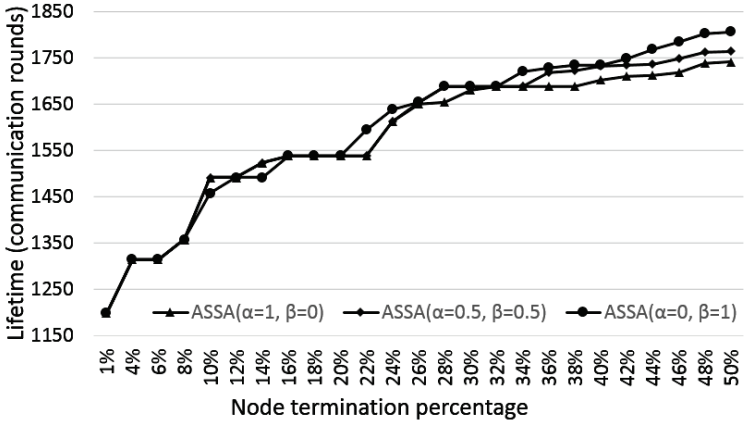

Figure 12: Simulated lifetime for the ASSA with different values of α and β.

thus extending its lifetime. Moreover, considering that a WSN may have unreliable sensor nodes, we developed two proactive fault tolerant routing algorithms. The EHAEC-1FT algorithm is based on EHAEC, and constructs two-connectivity routes that achieve one-node failure tolerance in WSNs. We also extended EHAEC-1FT to X-nFT for n-node failure tolerance using an arbitrary spanning tree routing algorithm X. We developed the ASSA to select active spare nodes in the network. By varying the impact factors, α and β, the ASSA can select active spare nodes that make the route more energy efficient or more fault tolerant.

We evaluated the effectiveness of our proposed algorithms in terms of two situations: energy efficiency is more critical, or fault tolerance is more critical. When not considering the fault tolerance, EHAEC extended the lifetime to more than 3.4 to 4.8 times that of a direct data transmission, and extended the lifetime to more than 1.8 to 2.2 times that of LEACH, in a 50-node network. When considering the fault tolerance, the proposed algorithms were more effective than $FGSS_k$. EHAEC-1FT with standby fault tolerance more effectively extended the WSN's lifetime than EHAEC, and EHAEC-1FT with active fault tolerance was less effective than EHAEC. EHAEC-1FT was more effective than EHAEC in terms of the fault tolerance, because it can tolerate the failure of one node. Moreover, our simulation results showed that the ASSA was more efficient than EHAEC-1FT, that large values of α produce a highly fault tolerant topology, and that large values of β increase the

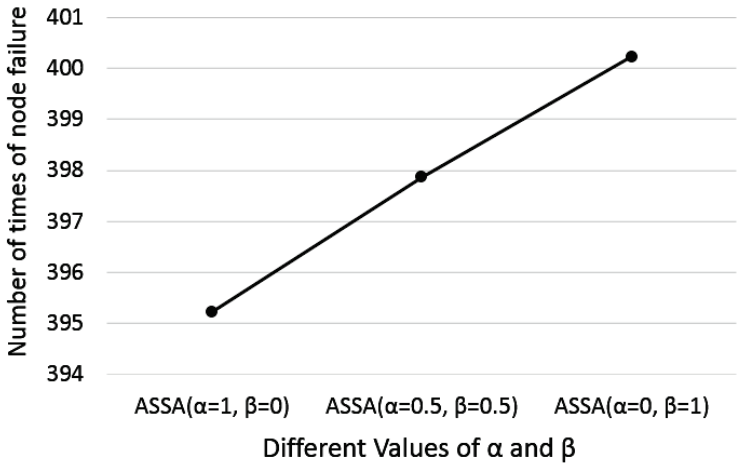

Figure 13: Simulated number of node failures for the ASSA with different values of α and β.

energy efficiency.

References

[1] Q. Zhao and Y. Nakamoto, "Algorithms for Reducing Communication Energy and Avoiding Energy Holes to Extend Lifetime of WSNs," IEICE Transactions, Vol. E97D, No.12, pp.2995-3006, Dec. 2014.

[2] Q. Zhao and Y. Nakamoto, "Routing Algorithms for Preventing Energy Holes and Improving Fault Tolerance in Wireless Sensor Networks," Proc. 2nd International Symposium on Computing and Networking Workshop (7th International Workshop on Autonomous Self-Organizing Networks), pp. 278-283, Shizuoka Japan, Dec. 2014.

[3] 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," Computer Networks 58, pp. 254-283, 2014.

[4] S. Olariu and I. Stojmenovic, "Design Guidelines for Maximizing Lifetime and Avoiding Energy Holes in Sensor Networks with Uniform Distribution and Uniform Reporting," Proc. IEEE International Conference on Computer Communications, pp. 1-12, Apr. 2006.

[5] M. Ding, X. Cheng and G. Xue, "Aggregation tree construction in sensor networks," IEEE 58th Vehicular Technology Conference, Vol. 4, pp. 2168-2172, Oct. 2003.

[6] N. Kimura and S. Latifi, "A Survey on Data Compression in Wireless Sensor Networks," Proc. the International Conference on Information Technology: Coding and Computing, Vol. 2, pp. 8-13, April 2005.

[7] H. Karl and A. Willig, "Protocols and Architectures for Wireless Sensor Networks," John Wiley, 2005.

[8] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," Proc. the 33rd Hawaii International Conference on System Sciences, Vol. 8, Jan. 2000.

[9] S. Lindsay and C. S. Raghavendra, "PEGASIS: Power-Efficient GAthering in Sensor Information Systems," Aerospace Conference Proceedings, Vol. 3, pp. 3-1125 - 3-1130, 2002.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to algorithms," 3rd ed., The MIT Press, 2009.

[11] X. Wu, G. Chen and S. K. Das, "Avoiding Energy Holes in Wireless Sensor Networks with Nonuniform Node Distribution," Parallel and Distributed Systems, IEEE Trans. Vol. 19, No. 5, pp. 710-720, May 2008.

[12] J. Lian, K. Naik and G. B. Agnew, "Modeling and enhancing data capacity in wireless sensor networks," IEEE Monograph on Sensor Network Operations, IEEE Press, 2004.

[13] J. C. Hou, N. Li and I. Stojmenovic, "Topology Construction and Maintenance in Wireless Sensor Networks," Handbook of sensor networks: algorithms and architectures, John Wiley, pp. 311-341, 2005.

[14] N. Li and J. C. Hou, "FLSS: a fault-tolerant topology control algorithm for wireless networks," Proc. the 10th ACM Annual International Conference on Mobile Computing and Networking, pp. 275-286, 2004.

[15] X. Y. Li, P. J. Wan, Y. Wang and C. W. Yi, "Fault Tolerant Deployment and Topology Control in Wireless Networks," Proc. the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 117-128, 2003.

[16] M. Jorgic, I. Stojmenovic, M. Hauspie and D. Simplot-Ryl, "Localized Algorithms for Detection of Critical Nodes and Links for Connectivity in Ad Hoc Networks," Proc. the 3rd IFIP Mediterranean Ad Hoc Networking Workshop, pp. 360-371, 2004.

[17] A. Kashyap, S. Khuller and M. Shaman, "Relay placement for higher order connectivity in wireless sensor networks," I Proc. IEEE International Conference on Computer Communications, Vol. 17, No. 3, 2006.

[18] W. Zhang, G. Xue and S. Misra, "Fault-tolerant node placement in wireless sensor networks: problems and algorithms," Proc. IEEE International Conference on Computer Communications, pp.1649 1657, 2007.

[19] Z. Yun, X. Bai, T. H. Lai and W. Jia, "Optimal deployment patterns for full coverage and connectivity wireless sensor networks," IEEE/ACM Transactions on Networking 18(3), pp.934 947, 2010.

[20] S. Vaidya and M. Younis, "Efficient failure recovery in wireless sensor networks through active spare designation," Proc. IEEE 6th International conference on Distributed Computing in Sensor Systems Workshops, pp. 1 6, 2010.

[21] K. Akkya, A. Thimmapuram, F. Senel and S. Uludag, "Distributed recovery of actor failures in wireless sensor and actor networks," Proc. IEEE Wireless Communication and Networking Conference, pp.2480 2485, 2008.

[22] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: A survey," Communications Surveys and Tutorials IEEE, Vol. 8, No. 4, pp. 48-63, 2006.