HKU CS Tech Report TR-2018-01

Adler: A Resilient, High-Performance and Energy-Efficient UAV-Enabled Sensor System

Dongda Li
The University of Hong Kong

Yuexuan Wang
Zhejiang University
The University of Hong Kong

Zhaoquan Gu
Guangzhou University
The University of Hong Kong

Tong Shen
Zhejiang University
The University of Hong Kong

Tianhao Wei
Zhejiang University

Yongqin Fu
Zhejiang University
The University of Hong Kong

Heming Cui
The University of Hong Kong

Mingli Song
Zhejiang University

Francis C. M. Lau
The University of Hong Kong

ABSTRACT

Sensor networks have been widely studied, but implementing a real system that achieves resilience, high performance and energy efficiency simultaneously is still a challenging task. We present Adler, a real unmanned aerial vehicle (UAV) enabled sensor system that can meet that challenge for general applications. We demonstrate Adler in three fundamental applications: localization, gathering and network reconfiguration. Evaluation results validate Adler's ability to achieve resilience, high performance and energy efficiency. Using Adler's application program interfaces (APIs), it is easy to evaluate algorithms for a pure sensor system or a UAV-enabled sensor system. Adler's source code is available at https://github.com/hku-systems/adler.

1 INTRODUCTION

Sensor networks are widely adopted in monitoring tasks including natural disaster[5], pollution [22], and industrial equipment [28]. A set of sensor nodes is deployed into a target area to sense for specific information, such as collecting temperature or humidity over agricultural lands. Sensed data can be aggregated to a base station by two common methods: a composed multi-hop network among the sensors[31], or a mobile vehicle that traverses the area to fetch data from nearby sensors[34].

Unfortunately, these methods cannot achieve resilience, high-performance and energy-efficiency at the same time. For instance, in investigating volcanic information, it is dangerous to drive a vehicle to collect data, and even an autonomous vehicle is limited by harsh conditions[40]. A multi-hop network shows poor performance in collecting data because many collisions and interference problems lead to package loss in a dense network, many isolated nodes find it difficult to send data to the base station if sensors are sparsely distributed[3]. In addition, a multi-hop network suffers from the energy-hole problem[43] that some nodes run out of energy faster when they have more packages to transmit.

The rapid development of unmanned aerial vehicle (UAV) technology makes it optimistic for collecting sensors' data through a UAV-enabled system[17, 20, 38, 39, 46], which has three major advantages compared to a multi-hop network or a mobile vehicle method. First, UAVs are able to reach harsh places, which makes the system more flexible for real-life applications and tolerate harsher conditions compare to a mobile vehicle method. Once some nodes run out of energy, a multi-hop network might break down, while a UAV-enabled system still works. In short, a UAV-enabled sensor system can be much more resilient than existing sensor networks without UAVs. Second, UAVs can fly close to a sensor node to collect data as this has lesser interference than a multi-hop network. Compared to a mobile vehicle, a UAV is faster and has better communication quality with the sensor. We evaluated the received signal strength indicator (RSSI) between a UAV and a sensor in real experiments, the RSSI is much higher when the UAV hovers in the air than it is on the ground. Third, a UAV-enabled sensor system could prolong sensors' lifetime, because UAVs may be recharged when they fly back to the base station while the sensors are difficult to recharge in harsh environments.

However, to the best of our knowledge, there exists no real implementation of such a system that achieves good performance for general applications. The existing UAV-enabled sensor systems are mainly evaluated by extensive simulations[17, 25, 39, 46]; only a few of them implement a real system for a specific task[29, 36]. For example, UAVs are utilize to gather sensed marine-coastal environmental data in [36]. The implemented systems cannot be resilient to more general applications since they lack a highly unified system framework. Some works use UAVs carrying sensors, such as cameras, for monitoring[21, 45], but they require a large number of UAVs to cooperate simultaneously.

In this paper, we present Adler[1], a real UAV-enabled sensor system that achieves resilience, high performance and energy efficiency concurrently, which builds on a unified framework for general applications. By utilizing Adler's application program interfaces (APIs), it is easy to evaluate any algorithm or run any application on a pure sensor network or a UAV-enable sensor system. We implemented three fundamental applications: localization, gathering, and network reconfiguration as examples. First, to get a sensor's three-dimension (3D) position if global position system (GPS) data

[1] Adler means eagle and it is the symbol of John the Evangelist who preaches and distributes sermons efficiently.

is unavailable or inaccurate, we design a UAV-based localization method, which improves localization accuracy compared to methods through a multi-hop network or a mobile vehicle (as anchor nodes on the ground). Second, to gather data efficiently, we propose a UAV-based method that uses the minimum number of hexagons to cover sensors, designs flight trajectory and gathers sensors' data by UAVs. This method reduces gathering latency compared to a mobile vehicle method. Third, to update parameter settings or switch between different applications/tasks, we present a UAV-based reconfiguration method that utilizes over-the-air (OTA) programming to reconfigure sensors through one-hop communication, which reduces package loss probability compared to a multi-hop OTA programming method.

We implemented Adler with DJI M100 UAVs and EZ240 Sensor nodes in real experiments, and we conducted extensive simulations for large-scale sensors. Adler improves localization accuracy of 20 sensors by reducing 78.4% root-mean-square error (RMSE) compared to methods by multi-hop networks or mobile vehicles. Adler achieves about 10% higher package receiving ratio compared to notable mobile sink methods for gathering application. Adler reduces sensors' average energy consumption by about 80% compared to multi-hop based methods. When the number of sensor nodes increases or some nodes run out of energy, Adler is more resilient and holds better performance than the state-of-the-art methods.

The contributions of Adler are threefold. First, Adler is easy to use by utilizing APIs. Many existing algorithms for sensor networks can be evaluated in our testbed via Adler. Second, Adler achieves resilience, high performance and energy efficiency at the same time. Thrid, We demonstrate three fundamental applications of Adler and evaluation results validate that Adler greatly improves these applications' performance in real scenario.

The paper is organized as follows. We introduce related works in the next section. Design of Adler is presented in Section 3, and we implement three fundamental applications: localization, gathering, and network reconfiguration in Sections 4-6 respectively. The evaluation results are provided in Section 7 and we conclude the paper in Section 8.

2 RELATED WORK

There are several existing UAV-enabled sensor systems, but they are only simulated for evaluation. For example, UAVs are utilized as mobile data collectors in a sensor system in [17, 20, 25, 29, 36, 39, 46]. However, most of them are evaluated by simulations, which are apart from real systems. For example, interference could cause connection condition change dynamically[1], but it is not considered in these simulations.

Many localization methods are proposed for a sensor system but these methods incur low accuracy in the real experiments. Localization methods by a multi-hop network measure radio signal parameters, such as the time of arrival (TOA), the angle of arrival (AOA) or the received signal strength (RSS), but they consume much energy and incur accumulative errors[2, 13]. Localization methods by ground anchor nodes, such as mobile vehicles, are interfered by ground-ground communication (two nodes on the ground) accuracy, which has lower resolution compared to air-ground communication (a UAV in the air)[14, 15]. UAVs are utilized to locate sensor nodes through RSS measurement in [38], but they assume the measured values and computed distances are precise. Most of them do not consider the measuring error in a real environment.

Gathering protocols by a multi-hop network have the energy hole problem, such as SPIN[31]), LEACH[16], and EBRP [33]. Though LEACH and EBRP utilize clusters to reduce energy cost, they cannot solve the energy hole problem at all. Gathering by a mobile vehicle is limited by ground conditions, such as DAWN [34], and the link quality would be worse than UAV-based gathering methods due to obstacles and various ground-ground interference.

Network reconfiguration is commonly achieved by multi-hop flooding methods, such as DPMT [11] and BLU [41]), but these methods incur broadcast storm problem [37]. Notable methods, such as Deluge and Rateless Deluge [12, 19], utilize multi-hop OTA programming for reconfiguration. Mobile Deluge proposed in [47] solves the broadcast problem, where a human carrying a mobile firmware reconfigures sensors manually. However, this method is quite inefficient. Utilizing UAVs for reconfiguration would be a promising way and we present the method in the paper.

3 DESIGN OF ADLER

3.1 System Overview

As shown in Fig. 1, Adler consists of three important components: sensors that are deployed in a target area, UAVs that cooperate with sensors, and a base station for mission scheduling and management.

Specifically, a sensor senses information and communicates via a wireless channel. Each UAV is equipped with a powerful airborne computer for computation and has a rechargeable battery for flight and wireless communication. Note that, UAVs are equipped with high gain antenna that achieves larger communication range than the sensors and thus a UAV can cover a large target area. If a UAV's energy is insufficient, it can fly back to the base station to recharge.

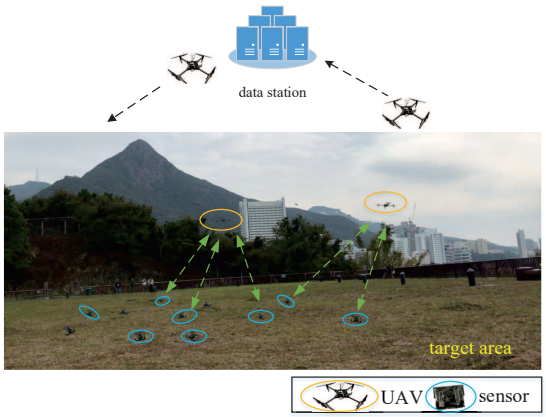


Figure 1: An overview of Adler

Sensor systems are capable of varied applications. We define an **application** as a specific task such as gathering data, broadcasting, etc. A **mission** is defined as an application in a target area with specific data, such as gathering humidity data in a farmland. A **firmware** is defined as an executive program for a sensor to run.

In Adler, a mission is sent from the base station to UAVs with input and expected output as follows:

- Input: a target area, targeted sensors, and application data such as firmware to be upgraded;

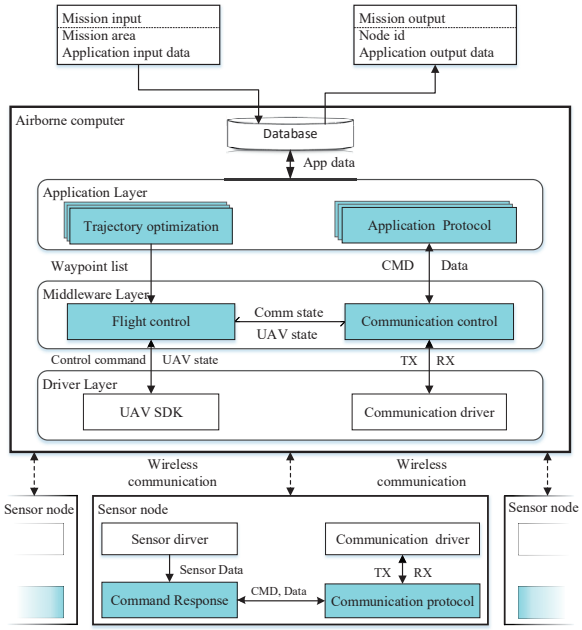


Figure 2: Architecture of Adler

- Output: collected data from sensors or other application results for the mission.

To carry out a new mission or to evaluate existing application algorithms, Adler simplifies the process as follows. First, implement an application algorithm by utilizing Adler's APIs and compile it to a firmware. Second, the base station sends the firmware to UAVs which would fly to re-program targeted sensors. Adler designs flight trajectory for each UAV to reach targeted sensors efficiently, and upgrades the firmware to sensors via OTA programming method. Finally, UAVs collect data to the base station or sensors run the firmware to evaluate application algorithms.

3.2 Adler Architecture

We propose a unified architecture for Adler in Fig. 2. A UAV has three main layers: *application layer* records the application protocol and designs its trajectory, *middleware layer* coordinates its motion and communication, and *driver layer* includes UAV's software development kit (SDK) and communication drivers. We designed both application layer and middleware layer for UAVs in Adler. A sensor node has two main layers: *application layer* contains application protocols and sensing functions, and *driver layer* includes sensor and communication drivers.

3.2.1 ***Application Layer of a UAV.*** According to the received mission, application layer executes the following subtasks in order: gets mission data from the base station, optimizes flight trajectory, obtains communication data and stores application data to the base station. There are two main components in the layer:

Trajectory Optimization is to calculate the UAV's trajectory on the basis of target area and targeted sensors' positions. Specifically, it computes the coordinates and order of the UAV's waypoints. Most applications need the UAV to traverse targeted sensors quickly, a good trajectory optimization method can reduce mission latency.

Application Protocol is the core of executing an application, which contains: 1) getting input from the base station; 2) generates data and application firmware to be sent; 3) broadcasts wakeup signals to sensor nodes; 4) schedules communication process; 5) obtains application data and send the output to the base station.

3.2.2 ***Middleware Layer of a UAV.*** The middleware layer consists of two main components. *Flight control* is to generate flight control commands according to waypoints from trajectory optimization. It switches flight state between flying and hovering. *Communication Control* is to maintain communication state with sensor nodes and other UAVs, which switches state between sleeping and working.

Coordinating flight control and communication control is a vital task and we propose state machine of middleware layer as Fig. 3. In the beginning, the UAV takes off and flies to the first mission waypoint generated by application layer. When it arrives, it publishes an arrived message and keeps hovering by flight control component. The communication control component switches to working state and communicates with sensor nodes or other UAVs when it receives the arrived message; when communication is finished, it publishes a communication finished message and switches to sleeping state. Once flight control component receives communication finished message, it calculates the energy (denote as E_b) needed to fly back to the base station. If remaining energy is less than $E_b + \delta$ or current waypoint is the last one, the UAV flies back for landing, where δ is maximum energy cost to hover for communication at a waypoint; otherwise, flight control component switches to flying state and the UAV flies to next mission waypoint.

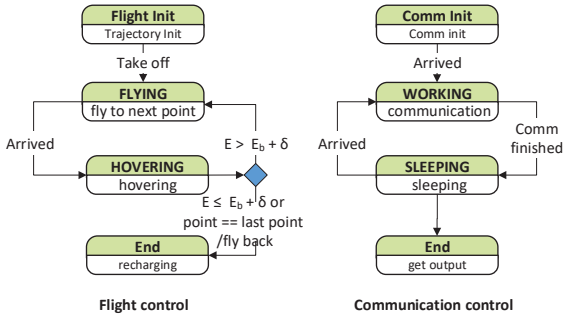


Figure 3: State machine of middleware layer

3.2.3 ***Application Layer of a Sensor.*** There are two main components in application layer of a sensor node: *communication protocol* schedules a sensor's duty cycle and communication with a UAV (in correspond to application protocol of a UAV) or other sensors; *command response* component receives commands from communication protocol component and responses to the commands, such as modifying duty cycle or getting sensed data. Since a sensor node has limited storage and memory, it cannot store different firmwares. In Adler, we design a UAV-based OTA programming method to upgrade and re-program a sensor's firmware for a new application or a new mission (please refer to Section 6).

3.2.4 ***Driver Layer of a UAV and a Sensor.*** A UAV's driver layer consists of UAV SDK and communication drivers, while a sensor's driver layer consists of sensors drivers and communication drivers. Both layers are implemented by manufactures. Communication drivers include basic network layer, physical layer, and data

link layer in Open Systems Interconnection model (OSI model). For a sensor, the communication driver contains more modules such as power management and embedded operation system (TinyOS [24], Contiki [9], etc). Sensor driver consists of basis interfaces such as open, measure, configure, close, etc.

3.3 Adler API

Table 1 shows Adler's APIs to simplify the usage. *Trajectory and flight APIs* provide control of a UAV's motion and feedback of the UAV's state; *application APIs* are responsible for UAVs' cooperations with sensors; *communication APIs* are designed for data transmission among UAVs and sensors, such as a UAV communicates with other UAVs or sensors, and a sensor communicates with other sensors; *sensor command response APIs* imply sensors' actions to commands.

Table 1: Adler API

Function	Description
UAV: Trajectory & Flight API	
wPointList getTraject(nodeSet)	calculate waypoint list according to nodes position
getUavStat(state)	get UAV state, including UAV's position, remaining power, etc.
uavFlyTo(wayPoint)	UAV flies to a specific waypoint and hovers there
UAV: Application API	
nodeSet detectSensor()	detect sensor nodes within communication range
upgrade(firmware)	upgrade sensor's firmware (such as compiled algorithms) via OTA
scanRssi(channel)	scan radio channel RSSI value
gathering(nodeSet)	gather data from sensors
sendCommand(nodeSet)	UAV send command to sensors
UAV & Sensor: Communication API	
unicast(nodeId,data)	send data to sensor or UAV
multicast(nodeSet,data)	send data to a group of destination
Sensor: Command Response API	
commandParse (command, len)	parse command from UAV starts the response of command
load(firmware)	load new firmware and restart

Listing 1: An example of upgrading sensors' firmware

```
wPointList = getTraject(nodeSet);
while(uavState.power > flyBackPower \
&& (wPointList.next != NULL) {
  uavFlyTo(wPointList.next);
  if(detectSensor().length != 0)
    upgrade("firmware.elf"); //OTA upgrade
  getUavStat(uavState);
}
uavFlyTo(station);
```

Listing 1 shows an example to upgrade a set of sensors by one UAV via OTA programming. The program first gets waypoint list of a mission; then the UAV flies to waypoints one by one to upgrade sensors' firmware until the mission is finished or UAV's energy is exhausted. At last, the UAV flies back to the base station.

Listing 2 shows a gathering algorithm we implemented by utilizing Adler's APIs. Compiled file of Listing 2 is the firmware to be upgraded on sensors. By Line 6 of Listing 1 (upgrade("firmware.elf")), the firmware can be upgraded through OTA programming.

Listing 2: An gathering example running on a sensor

```
sensor_thread(){
  while(1){
    sleep(gather_interval); //save energy
    unicast(nodeId,sensorData);
  //send sensor data to other sensors or UAVs
  }
} //the codes are compiled to firmware.elf
```

On the basis of Adler's APIs, it is very easy to implement and evaluate varied applications. We demonstrate three fundamental applications: localization, gathering and network reconfiguration in Sections 4-6 respectively.

4 THE LOCALIZATION APPLICATION

Localization is a fundamental step in a sensor system [1, 2, 13, 15, 23] if sensors' position information is unavailable or inaccurate. In Adler, we implement localization application by a UAV-based method. We formulate the localization problem as follows:

Problem 1. *Considering a target area with N deployed sensors at locations $S = \{S_1, S_2, \ldots, S_N\}$, in which $S_i = ({S_x}^i, {S_y}^i, {S_z}^i)$ referring to (x, y, z) coordinates respectively. Localization problem is to determine all locations' positions in S.*

4.1 Overview of UAV-based Localization

The UAV-based localization method consists of three modules: *grid module* divides a target area into grids; *trajectory module* determines UAVs' flight trajectory to traverse all sensors; *localization module* calculates each sensor's location coordinates by Received Signal Strength Indicator (RSSI).

RSSI is commonly used to measure the distance between a sender and a receiver, which follows the equation [44]:

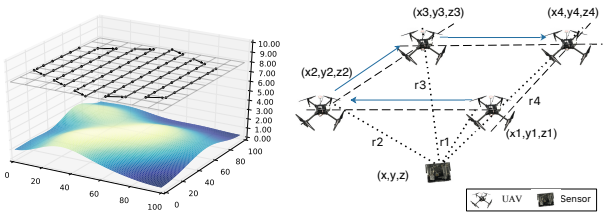
$$RSSI = A + 10nlog_{10}(d); \quad (1)$$

where A represents the received signal strength (dBm) at where the distance is $1m$ away from the sender, n represents the pass loss exponent depending on environment, and d represents the distance to measure. In Adler, UAVs can call the API *scanRssi(channel)* to collect RSSI at different positions. According to our evaluations, RSSI is much more accurate when UAVs hover in the air than they are on the ground. Therefore, a UAV-based localization method can achieve better accuracy than existing localization algorithms that use a mobile vehicle as anchor nodes on the ground.

4.2 Grid and Trajectory

In the localization process, the target area is first divided into grids and then UAVs traverses all grid vertices by designed flight trajectories. Actually, localization can be realized by even one UAV, whose flight trajectory is depicted in Fig. 4(a). Each vertex serves as a signal monitoring place. When a UAV arrives at a grid vertex, it keeps hovering until localization module is finished. Afterwards, it flies to next grid vertex by the designed trajectory. Note that, we choose appropriate grid length and fight height such that a sensor can communicate with a UAV locating at four grid vertices

successfully. If there are more UAVs, localization can be realized more efficiently.



(a) Grid and Trajectory (b) Localization

Figure 4: An example of localization application

4.3 Localization

When a UAV arrives at a grid vertex, it broadcasts a signal to nearby sensors and a sensor returns an acknowledgment signal containing its identifier (ID) and RSSI of the broadcasted signal. After receiving sensors' signals, the UAV records its current position and a list of ID and RSSI map. Due to page limits, we omit some details about receiving sensors' signals within grids. When the UAV traverses a grid's four vertices, it computes sensors' position coordinates by the following equations:

$$\begin{cases} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2=r_1^2 & \text{(2a)} \\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2=r_2^2 & \text{(2b)} \\ (x-x_3)^2+(y-y_3)^2+(z-z_3)^2=r_3^2 & \text{(2c)} \\ (x-x_4)^2+(y-y_4)^2+(z-z_4)^2=r_4^2 & \text{(2d)} \end{cases}$$

where (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4) are the coordinates of the four vertices of a grid (denote the vertices as v_1, v_2, v_3, v_4 respectively); r_1, r_2, r_3, r_4 are computed distances from a sensor (with the same ID) to four vertices by *RSSI* values; (x, y, z) is the sensor's coordinate to calculate. Note that, the designed trajectory in Fig. 4(b) ensures that every sensor in a grid can receive a broadcast signal from the grid's four vertices when the UAV traverses.

If computed distances $r_i, i \in [1, 4]$ are precise, the equations have exactly two solutions, one is the sensor's position while the other one is above the UAV. Since the UAV flies above sensors, it is easy to find the correct solution. However, computed distances are always inaccurate in practice and we present a position estimation algorithm to address the inaccuracy issue as Alg. 1. The idea behind the algorithm is: every three equations of Eqn. (2) have two solutions and the one with smaller z-coordinate is chosen; and then calculate the error compared to the other equation and select the solution with minimal error. We implemented the algorithm in Adler for real experiments and the results show that this method can achieve good localization accuracy.

4.4 Discussions

The UAV-based localization method has many advantages. As evaluated, RSSI between air-ground communication (a sensor and a hovering UAV) is more accurate ground-ground communication (two devices on the ground). Thus, the UAV-based method suffers from less interference from ground communication and greatly improves localization accuracy than these methods through a mobile vehicle or anchor nodes on the ground. As evaluated in Section 7.3, Adler reduces 78.4% root-mean-square error (RMSE) compared to ground anchor method. In addition, detected RSSI values are sometimes inaccurate, we address the real problem which is ignored in many existing methods.

Algorithm 1 Position Estimation Algorithm

1: For any three equations in Eqn. (2), compute the solutions and choose the appropriate one with smaller z-coordinate as:
2: Equation (2a) (2b) (2c) $\rightarrow p_a : (x_a, y_a, z_a)$;
3: Equation (2a) (2b) (2d) $\rightarrow p_b : (x_b, y_b, z_b)$;
4: Equation (2a) (2c) (2d) $\rightarrow p_c : (x_c, y_c, z_c)$;
5: Equation (2b) (2c) (2d) $\rightarrow p_d : (x_d, y_d, z_d)$;
6: For p_a and vertex v_4, calculate the Euclidean distance between them as d_{a1} and compute distance by RSSI at v_4 and Eqn. (1) as d_{a2}. Calculate $\delta_a = |d_{a1} - d_{a2}|$;
7: Repeat the process in Line 6 for p_b and v_3, p_c and v_2, p_d and v_1 to get δ_b, δ_c, δ_d respectively;
8: Select smallest $\delta_i (i = a, b, c, d)$ and set corresponding p_i as the sensor's position.

5 THE GATHERING APPLICATION

Gathering is an important application for a sensor system. The base station gathers sensors' data, such as temperature [30], or humidity [4] for further analysis and decisions. In Adler, we implement gathering application by a UAV-based method, which has obvious advantages than traditional methods[16]. We formulate gathering problem in a 2-dimension target area as follows:

Problem 2. *Considering a 2D target area and N targeted sensors at locations $S = \{S_1, S_2, \ldots, S_N\}$, in which $S_i = (S_x^i, S_y^i)$. Gathering problem is to collect all sensors' data to the base station.*

5.1 Overview of UAV-Based Gathering

We present a UAV-Based gathering method in Adler, in which UAVs are dispatched to collect each sensor's data at location $S_i \in S$. The method consists of three modules: *hexagon covering module* divides targeted sensors into hexagons, and a UAV hovering at a hexagon's center point can collect all sensors' data within the hexagon; *trajectory module* determines UAVs' flight trajectories to traverse all hexagons' center point; *gathering module* designs data collecting scheme from sensors which a sensor calls the API *unicast(nodeId, data)* of Section 3.3. The method can work even if there is only one available UAV and we mainly introduce the method to gather sensors' data by a single UAV. Obviously, more UAVs can gather data more efficiently and the difference is: the flight trajectories should be designed carefully.

5.2 Hexagon Covering

If sensors are distributed sparsely in a target area, a UAV can visit sensors one by one. Considering a general scenario which more sensors are deployed at a sensitive area, we first divide sensors into K clusters by adopting clustering algorithms[7, 32]. For each cluster, we use a minimum number of hexagons to cover it. Technically, assume a sensor's communication range is r_u and a UAV flies at a height of h, we use hexagons with side length $a = \sqrt{r_u^2 - h^2}$ to cover each cluster. Suppose a cluster has n sensors with positions $S = \{S_1, S_2, \ldots, S_n\}$, we propose a minimum hexagon covering algorithm as Alg. 2. The idea behind the algorithm is: first find

an arbitrary hexagon covering scheme for a cluster, which can be done easily because hexagons can fully cover an arbitrary area without overlapping. Then move the scheme along x-axis, or y-axis, or rotate by appropriate angles, compare the number of needed hexagons for these variations and compute the optimal hexagon covering scheme.

Algorithm 2 Minimum Hexagon Covering Algorithm

1: Find an arbitrary hexagon covering scheme as $C = \{H_1, H_2, \ldots, H_k\}$;
2: **for** Each hexagon H_i with center point h_i **do**
3: Find S_j covered by H_i such that it has largest distance to h_i;
4: Move C to $\mathcal{S}_i$ along x-axis and y-axis such that the left-bottom vertex of H_i coincides S_j;
5: Rotate C_i by $\theta \in (0, \frac{\pi}{3}]$ according to left-most position and bottom-most position;
6: Record the number of hexagons to cover S every time;
7: **end for**
8: Choose the covering scheme with the minimum number of hexagons.

Fig.5 shows an example of clustering and hexagon covering scheme, which different colors represent different clusters and hexagons with solid lines are generated scheme. We evaluate Alg. 2 is Section 7 and it greatly reduces the number of hexagons to cover the targeted sensors as Fig. 8.

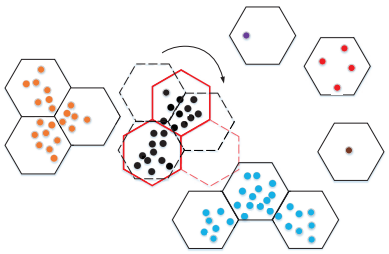


Figure 5: An example of clustering and hexagon covering. The black hexagon is the original coverage, the red hexagon is the coverage after rotation, and the red dotted hexagon is removed after rotation.

5.3 Trajectory

Once the hexagons covering scheme is determined, we denote the center point of each hexagon as $P = \{p_1, p_2, \ldots, p_M\}$, and we need to design flight trajectory that traverses all points $p_i \in P$. We utilize existing algorithms for the Travelling Salesman Problem (TSP), in which a UAV starts from the base station (denote as p_0), traverses all points, and flies back to the station. There are many efficient algorithms, such as Genetic algorithm [35], Simulated Annealing algorithm [27], and Ant Colony Optimization (ACO) algorithm [10]. We choose the ACO algorithm in Adler to design UAV's trajectory. Note that a UAV has limited energy and denote the largest distance it can fly as Δ. We modify the ACO algorithm to find set of minimum cardinality $\{P_1, P_2, \ldots P_k\}$ where $P_i \subseteq P$, $P_i \bigcap P_j = \emptyset, i \neq j$, such that the flight distance of a UAV starting at p_0, traversing all points in set P_i, and flying back to p_0 is less than Δ.

5.4 Data Gathering

A sensor only needs to send its data to a UAV that is within communication range. In Adler, a sensor has three modes: sleep, listen, transmit; it switches between sleep and listen to save energy. When a UAV arrives at a center point of a hexagon, it hovers there and broadcasts a beacon message. Once a sensor receives a beacon message from the UAV, it switches to transmit mode and sends data to the UAV according to p-persistent CSMA protocol [31] (to reduce collision with other transmissions). Then, it returns to sleep or listen mode as normal. The UAV hovers at the point to receive data for a given time, and then it flies to next center point.

5.5 Discussions

The UAV-based gathering method has many advantages. Compared to data gathering through a multi-hop network, the UAV-based method could save a sensor's energy hugely which overcomes energy hole problem in a multi-hop network (as stated in Fig. 10 in Section 7.4). In addition, the UAV-based method suffers from less communication collision since a sensor only needs to communicate with the UAV. Compared to a mobile vehicle gathering method, the UAV-based method designs a better trajectory since it does not need to consider complicated ground conditions. In addition, a UAV has a faster speed to collect data. As evaluated in Section 7.3, Adler achieves 10% higher package receiving ratio in Fig. 9. The disadvantage of the method is: the UAV has to fly to reach sensors and the latency to collect all data is limited to flight distance and speed. Gathering through a multi-hop network could be faster, but has the instinctive energy problem. Actually, if more UAVs are utilized in gathering, each UAV is responsible for gathering sensors in a divided area and the composed network among UAVs could largely reduce gathering latency. It would be a promising way to combine a sensor network and a UAV network in future.

6 THE NETWORK RECONFIGURATION APPLICATION

Network reconfiguration is essential in a long-term sensor system, such as upgrading applications (firmwares) for different missions or updating some sensors' parameters. Conventional flooding methods incur broadcast storm problem[37], which waste energy and require massive retransmissions. In Adler, we implement network reconfiguration application by an UAV-based method. Similar to gathering, a single UAV is able to realize the application, which can fly close to a sensor and send data to it by point-to-point transmission. This method does not incur broadcast storm problem obviously. We formulate the network reconfiguration problem as follows:

Problem 3. *Considering a 2D target area and N targeted sensors $\{s_1, s_2, \ldots, s_N\}$, network reconfiguration is to update sensor s_i's parameter or to upgrade its running application.*

6.1 Overview of UAV-Based Reconfiguration

We present a UAV-based reconfiguration method in Adler, in which a single UAV is dispatched to update each sensor's data. Reconfiguration is some kind opposite to gathering, since a UAV sends data to a sensor in reconfiguration while a sensor sends data to a UAV for gathering. Similarly, the method consists of three modules: *hexagon covering module* divides target sensors into hexagons; *trajectory module* determines UAV's flight trajectory to traverse all

hexagons' center point; and *reconfiguration module* updates sensors' parameters or upgrades its firmware. The UAV calls the API *multicast(nodeSet, data)* to reconfigure sensors, and a sensor calls the API *load(firmware)* of Section 3.3.

Actually, hexagon covering module and trajectory module are the same as those in gathering application (Section 5). Hence, we focus on network configuration design.

6.2 Network Reconfiguration

Network reconfiguration includes parameter update and firmware upgrade. For example, the base station tries to modify a sensor's duty cycle setting, it is one kind of parameter update; if the base station is to evaluate a new sensor algorithm, it would upgrade a sensor's executive program, which is one kind of firmware upgrade.

Parameter update: in order to configure parameters smartly, a sensor node initializes a '.ini' file in its flash memory, which contains frequently-used parameters and settings. A sensor node reads the '.ini' file when it starts to work or receives a new '.ini' file. When a UAV hovers at the center point of a hexagon, it broadcasts a beacon (parameter update) signal including a target sensor's ID. A sensor tunes to listen mode periodically as stated in Section 5.4; if it receives a parameter update signal containing its ID, it transmits with an acknowledgment signal and waits for the update. Then, the UAV sends corresponding '.ini' file to the sensor; once it is finished, the sensor saves the file in its flash memory and reboots, while the UAV sends another parameter update signal for other sensors to continue the process.

Firmware upgrade: we use OTA programming to upgrade a sensor's firmware, which sends the firmware to a sensor node and makes the firmware run. The firmware is stored in form of an Extensible Linking Format (ELF) file, and we modify Deluge [19], a well-known data dissemination protocol to realize transmission between a UAV and a target sensor. When the sensor receives the ELF file, it decodes the file, loads to its code space and run it. The difference with parameter update is: a parameter file is of small size which can be transmitted directly, while the firmware is of much larger size which needs to be transmitted by an efficient protocol ensuring file completeness and reliability.

6.3 Discussions

The UAV-based network configuration method has many advantages. Conventional methods utilize a multi-hop OTA programming method to upgrade the firmware, such as Deluge, and Rateless Deluge [12]. However, these methods cost much energy of a (relay) sensor, since it has to forward the firmware to other sensors hop by hop. The firmware is of large size and multiple retransmissions are needed if the collision exists or a part of the file gets lost. The UAV-based method greatly reduces energy cost and package loss ratio. As evaluated in Section 7.4, Adler reduces sensors' average energy consumption by about 80% as Fig. 11. In addition, this method allows updating different parameters for different sensors. For example, to run a clustering-based gathering algorithm[33], the base station could generate clusters in a centralized way, and sends a UAV to update cluster information to sensors with different roles, such as some sensors receive a parameter as 'cluster-head', while some get a parameter as 'cluster-member'. Existing clustering methods consume much energy and the generated clusters are only a local optimal solution.

7 EVALUATION

7.1 Evaluation Setup

The design of Adler supports multiple UAVs for general applications. Since current three implemented applications only require one single UAV, we evaluate the applications, localization, gathering and network reconfiguration, by three UAVs working independently with 20 sensors for each application. We also evaluate Adler through simulations for a large scale sensors set.

7.1.1 Experiment Platform. We use three DJI M100 UAVs[2] and 60 EZ240[18] in the experiments. Each UAV carries an airborne computer with the Intel NUC Kit NUC7I5BNK[3], which contains an Intel Core i5-7260U processor with 8GB RAM and 240G SSD. The EZ240 sensor nodes uses MSP430F1611 as the microprocessor and it is equipped with 1K RAM, 48K ROM and 1 M Flash. IEEE 802.15.4 protocol is utilized as the communication protocol.

The software of the airborne computer is running on Ubuntu 16.04 Operating System (OS), and we build the development environment based on the Robot Operating System (ROS) and DJI Onboard SDK. The sensors are running Contiki OS with Contiki MAC[8] the Radio Duty Cycle Control module and CSMA [31] as media access control protocol.

In the experiments, we set UAVs' flying speed as $8m/s$, hovering height as $5m$ and assume the UAV hovers at a waypoint for $5s$ to collect or distribute data. For each application, we evaluate Adler by one UAV and 20 sensors.

7.1.2 Localization. To compute the value of A in Eqn. (1), we calibrate RSSI at $1m$ away position. Then, we implement the UAV-based localization algorithm in Adler. The UAV flies by a predetermined trajectory and calculates sensors' positions according to RSSI values of sensors. We compare our method with localization methods by ground anchor nodes[14].

7.1.3 Gathering. We implement the UAV-based gathering method in Adler for a real experiment. We evaluate several notable algorithms (SPIN[31], DAWN[34], MINT[42], LEACH[16], EBRP[33]) on 20 sensors that are distributed randomly in a $64m \times 64m$ area. To evaluate the reliability of the system, packages are not retransmitted if they get lost in transmission.

According to the UAV-based gathering method, we divide the target area into clusters based on sensors' positions. For each cluster, we compute the minimum hexagon covering scheme by Alg. 2 and design corresponding trajectory. In our simulations, we set the side length of a hexagon as $15m$.

7.1.4 Reconfiguration. In our experiments, we implement the UAV-based reconfiguration method in Adler, including parameter updating and firmware upgrading. To update parameters, we sent 100 packages to all sensors and each package's size is 128 bytes; to upgrade firmware, we implemented OTA by modifying Deluge and sent a LED blink code of size 1274 bytes to all sensors for evaluation.

[2] https://www.dji.com/cn/matrice100

[3] https://www.intel.com/content/www/us/en/products/boards-kits/nuc/kits/nuc7i5bnk.html

In simulation, we choose BLU[41] and DPMT[11] for comparison. The cost energy of a sensor is calculated as $E_i = \frac{\lambda}{\psi(d)}\omega_r$, where λ is data size of the configuration file.

7.2 Evaluation metrics

We introduce some evaluation metrics used in our experiments.

Localization accuracy. We use the root-mean-square error (RMSE) to evaluate the localization error:

$$RMSE = (\sum_{i=1}^{N} \sqrt{(x_i - \hat{x}_i)^2}/(N \times r_u)), \tag{3}$$

where x_i and $\hat{x}_i$ represents real and estimated coordinates for sensor i respectively. N is the number of targeted nodes and r_u is the communication range.

Receiving Package Ratio (RPR). We use packet reception probability model in [6] and introduce the metric *RPR*, which is the ratio of received packages to total packages that are sent, representing the throughput of a network.

Latency. Latency is defined as total time cost for the UAV to finish a mission, including traversing all targeted nodes, gathering or reconfiguring sensors, and flying back to the base station.

Network lifetime. We adopt the energy model in [26]. Network lifetime is defined as the elapsed time when the first sensor running out of energy, which is an important metric of energy efficiency. If the number of exhausted nodes increases, the network's RPR would be reduced.

Node Average Energy Cost (NAEC) computes average cost of all sensors when a specific number of packages have been sent.

7.3 Performance

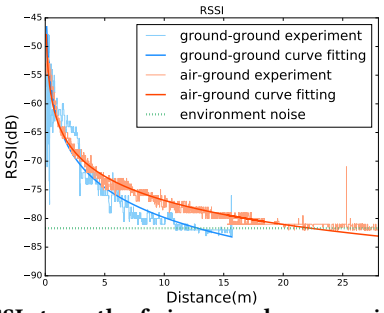
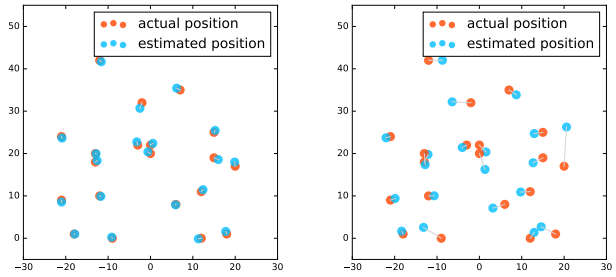


Figure 6: RSSI strength of air-ground communication is better than that of ground-ground communication. RSSI becomes invalid when it is below environment noise $-81.7dB$.

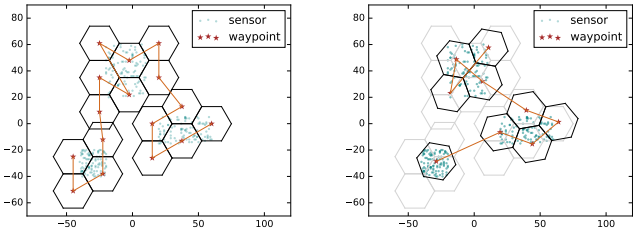
Localization. As shown in Fig. 6, air-ground communication (the UAV with a sensor) has a wider measuring range and suffers less noise/interference than ground-ground communication (two sensors or the UAV is on the ground). Therefore, UAV-based localization method can achieve better accuracy than the methods using ground landmark or anchor nodes. We measure Gaussian parameters of noise distribution in both methods by experiments. Air-ground experiment's Gaussian noise scale is about 0.9250 while ground-ground experiment's Gaussian noise scale is 6.0866. Hence, our method can achieve more accurate localization as shown in Fig. 7: RMSE of the UAV-based method is reduced 78.4% than that of the method using ground anchor nodes.



(a) UAV-based method (air-ground communication). RMSE is 0.02408

(b) Ground anchor method (ground-ground communication). RMSE is 0.11135

Figure 7: The UAV-based method greatly improves localization accuracy, which reduces 78.4% RMSE compared with ground anchor method.



(a) Hexagon covering scheme. Hexagon count is 16, gathering latency is 126s

(b) Hexagon covering scheme in Adler. Hexagon count is 9, gathering latency is 73s

Figure 8: Adler uses fewer hexagons to cover sensors and greatly reduces gathering latency.

Gathering & Reconfiguration. We evaluate the fundamental step: minimum hexagon covering algorithm of gathering and reconfiguration. We generate 100 sensors which are divided into three clusters. As shown in Fig. 8a, sensors are covered by 16 hexagons; while our method only utilizes 9 hexagons to cover them, as shown in 8b. Correspondingly, the gathering latency is reduced from 126s (in Fig. 8a) to 73s (in Fig. 8b); while the parameter updating latency for reconfiguration is reduced from 132s to 75s.

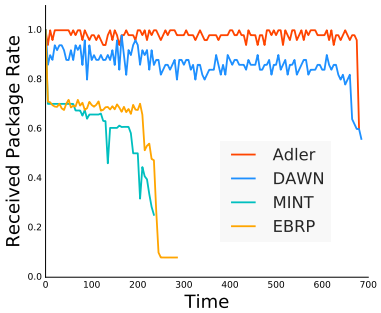


Figure 9: Adler greatly promotes RPR among all methods.

Gathering. We compare Adler with three gathering methods (DAWN, MINT, EBRP) and Adler achieves highest RPR among them during the whole procedure, as shown in Fig. 9. Both MINT and EBRP gather data through a multi-hop network and transmission

collisions lead to lower RPR than Adler; when time goes on (as x-axis), some sensors run out of energy and RPR of both methods go down quickly. DAWN gathers data by a mobile sink but it has lower RPR than Adler due to its need of road; wherever there is no road, sensors compose a multi-hop structure leading to lower RPR.

7.4 Energy Consumption

Gathering. A multi-hop gathering method faces energy hole problem and EBRP is a cluster-based method to alleviate it. We compare EBRP and Adler and simulations show Adler achieves about 10% higher receiving package ratio. After 1563 rounds of gathering, as Fig. 10 shows, sensors using Adler have very uniform residual energy value, while running EBRP, sensors that are close to the base station have less residual energy than sensors that are far away. This is because a multi-hop based method has its intrinsic deficiency and the UAV-based method could solve it with conciseness and effectiveness.

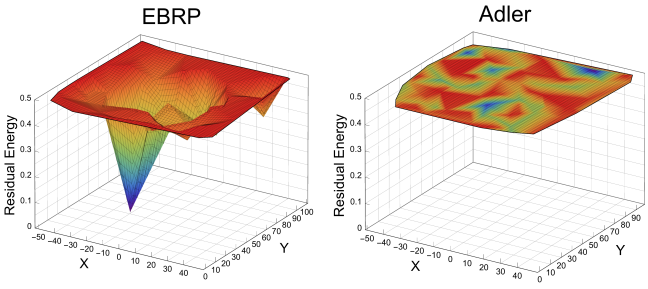


Figure 10: Adler solves energy hole problem which has more uniform residual energy distribution than EBRP.

Reconfiguration. We compare the UAV-based method with a multi-hop flooding method[11] in the experiment. As shown in Fig. 11, sensors that have different distances to the base station (as x-axis) have similar energy cost in Adler, while sensors that are close to the base station cost much more energy in the multi-hop flooding method. Notice that, in the multi-hop flooding method, sensors that are far away from the base station also consume more energy than Adler. This is because sensors have to rebroadcast the received messages in the multi-hop flooding method, while they only need to receive the message from the UAV in Adler.

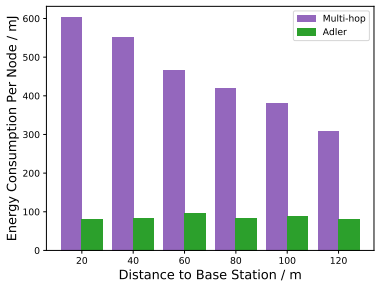
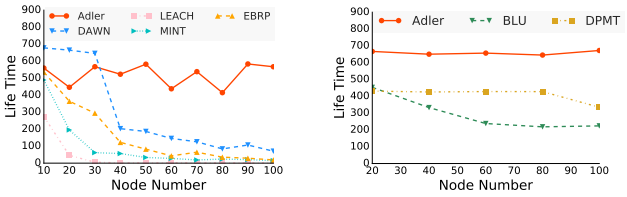


Figure 11: Adler consumes less energy than multi-hop method in reconfiguration.

7.5 Scalability

Gathering. We compare a sensor network's lifetime with four notable algorithms (DAWN, MINT, LEACH, EBRP) when the number of sensors increases in our simulations. As shown in Fig. 12a, Adler achieves the longest lifetime compared to others, and the lifetime in Adler is irrelevant to network size. This is because a sensor only needs to communicate once with the UAV in each round of gathering. The other methods have shorter lifetime because retransmissions in a multi-hop based method cost more energy.



(a) Gathering **(b) Reconfiguration**

Figure 12: Adler holds better scalability when the number of sensors increases.

Reconfiguration. We compare Adler with BLU and DPMT. Fig. 12b shows a similar phenomenon as gathering which Adler greatly prolongs a sensor network's lifetime.

7.6 Resilience

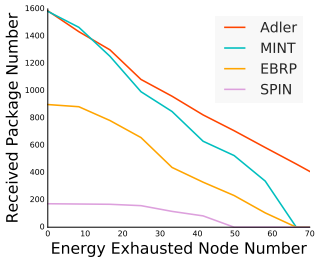


Figure 13: Adler is more resilient if some sensors run out of energy.

We evaluate network resilience on gathering application. When some nodes run out of energy (we tune them to sleep mode in the simulation) and Fig. 13 shows Adler still achieves the highest received package number among these methods. Similarly, the localization and network reconfiguration applications are also resilient as energy exhausted nodes increase.

8 CONCLUSION

In this paper, we implemented Adler, a real UAV-enabled sensor system which achieved resilience, high performance and energy efficiency at the same time for general applications. Three important applications: localization, gathering and network reconfiguration are realized in Adler by UAV-based methods. Adler greatly increased localization accuracy, reduced gathering latency, and prolonged network lifetime through experiments and simulations. Adler can be used to implement a general application easily and it has the potential to realize globalization in a distributed sensor system.

REFERENCES

[1] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. 2004. Link-level measurements from an 802.11 b mesh network. In *ACM SIGCOMM Computer Communication Review*, Vol. 34. ACM, 121–132.

[2] Joe Albowicz, Alvin Chen, and Lixia Zhang. 2001. Recursive position estimation in sensor networks. In *Network Protocols, 2001. Ninth International Conference on*. IEEE, 35–41.

[3] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. 2009. Energy conservation in wireless sensor networks: A survey. *Ad hoc networks* 7, 3 (2009), 537–568.

[4] Juan Carlos SuÃąrez BarÃşn and Marco Javier SuÃąrez BarÃşn. 2014. Application of SHT71 sensor to measure humidity and temperature with a WSN. In *Sensors*.

[5] Dan Chen, Zhixin Liu, Lizhe Wang, Minggang Dou, Jingying Chen, and Hui Li. 2013. Natural disaster monitoring with wireless sensor networks: a case study of data-intensive applications upon low-cost scalable systems. *Mobile Networks and Applications* 18, 5 (2013), 651–663.

[6] Zhuangbin Chen, Anfeng Liu, Zhetao Li, Young-June Choi, Hiroo Sekiya, and Jie Li. 2017. Energy-efficient broadcasting scheme for smart industrial wireless sensor networks. *Mobile Information Systems* 2017 (2017).

[7] Liliya Demidova, Yulia Sokolova, and Evgeny Nikulchev. 2015. Use of fuzzy clustering algorithms ensemble for SVM classifier development. *International Review on Modelling and Simulations* 8, 4 (2015), 446–457.

[8] Adam Dunkels. 2011. The contikimac radio duty cycling protocol. (2011).

[9] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. 2004. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 455–462.

[10] Daoxiong Gong and Xiaogang Ruan. 2004. A hybrid approach of GA and ACO for TSP. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*. 2068–2072 Vol.3.

[11] Song Guo and Oliver Yang. 2004. Multicast lifetime maximization for energy-constrained wireless ad-hoc networks with directional antennas. In *GLOBECOM'04. IEEE*, Vol. 6. IEEE, 4120–4124.

[12] Andrew Hagedorn, David Starobinski, and Ari Trachtenberg. 2008. Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes. In *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society, 457–466.

[13] Guangjie Han, Huihui Xu, Trung Q Duong, Jinfang Jiang, and Takahiro Hara. 2013. Localization algorithms of wireless sensor networks: a survey. *Telecommunication Systems* (2013), 1–18.

[14] Guangjie Han, Huihui Xu, Jinfang Jiang, Lei Shu, Takahiro Hara, and Shojiro Nishio. 2013. Path planning using a mobile anchor node based on trilateration in wireless sensor networks. *Wireless Communications and Mobile Computing* 13, 14 (2013), 1324–1336.

[15] Guangjie Han, Chenyu Zhang, Jaime Lloret, Lei Shu, and Joel JPC Rodrigues. 2014. A mobile anchor assisted localization algorithm based on regular hexagon in wireless sensor networks. *The Scientific World Journal* 2014 (2014).

[16] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*. IEEE, 10–pp.

[17] Dac-Tu Ho, Esten Ingar Grøtli, PB Sujit, Tor Arne Johansen, and João Borges Sousa. 2015. Optimization of wireless sensor network and UAV data acquisition. *Journal of Intelligent & Robotic Systems* 78, 1 (2015), 159.

[18] Tingpei Huang, Haiming Chen, Zhaoliang Zhang, and Li Cui. 2012. EasiPLED: Discriminating the causes of packet losses and errors in indoor WSNs. In *GLOBECOM, 2012 IEEE*. IEEE, 487–493.

[19] Jonathan W Hui and David Culler. 2004. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 81–94.

[20] Imad Jawhar, Nader Mohamed, Jameela Al-Jaroodi, and Sheng Zhang. 2014. A framework for using unmanned aerial vehicles for data collection in linear wireless sensor networks. *Journal of Intelligent & Robotic Systems* 74, 1-2 (2014).

[21] Tyler Kersnovski, Felipe Gonzalez, and Kye Morton. 2017. A UAV system for autonomous target detection and gas sensing. In *Aerospace Conference, 2017*.

[22] Kavi K Khedo, Rajiv Perseedoss, Avinash Mungur, et al. 2010. A wireless sensor network air pollution monitoring system. *arXiv preprint arXiv:1005.1737* (2010).

[23] Eunchan Kim, Sangho Lee, Chungsan Kim, and Kiseon Kim. 2010. Mobile beacon-based 3D-localization with multidimensional scaling in large sensor networks. *IEEE Communications Letters* 14, 7 (2010), 647–649.

[24] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. 2005. TinyOS: An operating system for sensor networks. *Ambient intelligence* 35 (2005), 115–148.

[25] Hanshang Li, Ling Wang, Shuo Pang, and Massood Towhidnejad. 2014. A cross-layer design for data collecting of the UAV-wireless sensor network system. In *Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on*. IEEE, 242–249.

[26] Anfeng Liu, Xin Jin, Guohua Cui, and Zhigang Chen. 2013. Deployment guidelines for achieving maximum lifetime and avoiding energy holes in sensor network. *Information Sciences* 230 (2013), 197–226.

[27] Yi Liu, Shengwu Xiong, and Hongbing Liu. 2009. Hybrid simulated annealing algorithm based on adaptive cooling schedule for TSP. In *Acm/sigevo Summit on Genetic and Evolutionary Computation*. 895–898.

[28] Kay Soon Low, Win Nu Nu Win, and Meng Joo Er. 2005. Wireless sensor networks for industrial environments. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*. IEEE.

[29] Andrew Mittleider, Brent Griffin, and Carrick Detweiler. 2016. Experimental analysis of a uav-based wireless power transfer localization system. In *Experimental Robotics*. Springer, 357–371.

[30] Yi Jen Mon, Chih Min Lin, and Imre J Rudas. 2012. Wireless Sensor Network (WSN) Control for Indoor Temperature Monitoring. *Acta Polytechnica Hungarica* 9, 6 (2012), 17–28.

[31] Asis Nasipuri, Jun Zhuang, and Samir R Das. 1999. A multichannel CSMA MAC protocol for multihop wireless networks. In *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, Vol. 3. IEEE, 1402–1406.

[32] Tina R Patil and SS Sherekar. 2013. Performance analysis of Naive Bayes and J48 classification algorithm for data classification. *International Journal of Computer Science and Applications* 6, 2 (2013), 256–261.

[33] Fengyuan Ren, Jiao Zhang, Tao He, Chuang Lin, and Sajal K Das Ren. 2011. EBRP: energy-balanced routing protocol for data gathering in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 22, 12 (2011), 2108–2125.

[34] ShaoJie Tang, Jing Yuan, XiangYang Li, Yunhao Liu, GuiHai Chen, Ming Gu, Jizhong Zhao, and Guojun Dai. 2010. DAWN: energy efficient data aggregation in WSN with mobile sinks. In *Quality of Service (IWQoS), 2010 18th International Workshop on*. IEEE, 1–9.

[35] Zhou Tao. 2008. TSP Problem Solution Based on Improved Genetic Algorithm. In *Fourth International Conference on Natural Computation*. 686–690.

[36] Carlos A Trasviña-Moreno, Rubén Blasco, Álvaro Marco, Roberto Casas, and Armando Trasviña-Castro. 2017. Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring. *Sensors* 17, 3 (2017), 460.

[37] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. 2002. The broadcast storm problem in a mobile ad hoc network. *Wireless networks* 8, 2/3 (2002), 153–167.

[38] Leandro A Villas, Daniel L Guidoni, and Jo Ueyama. 2013. 3d localization in wireless sensor networks using unmanned aerial vehicle. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*. IEEE.

[39] Chengliang Wang, Fei Ma, Junhui Yan, Debraj De, and Sajal K Das. 2015. Efficient aerial data collection with uav in large-scale wireless sensor networks. *International Journal of Distributed Sensor Networks* 11, 11 (2015), 286080.

[40] Geoffrey Werner-Allen, Jeff Johnson, Mario Ruiz, Jonathan Lees, and Matt Welsh. 2005. Monitoring volcanic eruptions with a wireless sensor network. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*. IEEE.

[41] Jeffrey E Wieselthier, Gam D Nguyen, and Anthony Ephremides. 2002. Energy-efficient broadcast and multicast trees in wireless networks. *Mobile networks and applications* 7, 6 (2002), 481–492.

[42] Alec Woo, Terence Tong, and David Culler. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 14–27.

[43] Xiaobing Wu, Guihai Chen, and Sajal K Das. 2008. Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *IEEE Transactions on parallel and distributed systems* 19, 5 (2008), 710–720.

[44] Jianqiao Xiong, Qin Qin, and Kemin Zeng. 2014. A distance measurement wireless localization correction algorithm based on RSSI. In *Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on*, Vol. 2. IEEE, 276–278.

[45] Zhenhui Yuan, Xiwei Huang, Lingling Sun, and Jie Jin. 2016. Software defined mobile sensor network for micro UAV swarm. In *Control and Robotics Engineering (ICCRE), 2016 IEEE International Conference on*. IEEE, 1–4.

[46] Cheng Zhan, Yong Zeng, and Rui Zhang. 2017. Energy-efficient data collection in UAV enabled wireless sensor network. *arXiv preprint arXiv:1708.00221* (2017).

[47] Xiaoyang Zhong, Miguel Navarro, German Villalba, Xu Liang, and Yao Liang. 2014. MobileDeluge: Mobile code dissemination for wireless sensor networks. In *MASS*. IEEE, 363–370.