electronics

MDPI

Article

Adaptive Clustering and Scheduling for UAV-Enabled Data Aggregation

Tien-Dung Nguyen [1], Tien Pham Van [1], Duc-Tai Le [2,*] and Hyunseung Choo [2,*]

[1] School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi 100000, Vietnam; dung.nguyentien2@hust.edu.vn (T.-D.N.); tien.phamvan1@hust.edu.vn (T.P.V.)
[2] Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea
* Correspondence: ldtai@skku.edu (D.-T.L.); choo@skku.edu (H.C.)

Abstract: Using unmanned aerial vehicles (UAVs) is an effective way to gather data from Internet of Things (IoT) devices. To reduce data gathering time and redundancy, thereby enabling the timely response of state-of-the-art systems, one can partition a network into clusters and perform aggregation within each cluster. Existing works solved the UAV trajectory planning problem, in which the energy consumption and/or flight time of the UAV is the minimization objective. The aggregation scheduling within each cluster was neglected, and they assumed that data must be ready when the UAV arrives at the cluster heads (CHs). This paper addresses the minimum time aggregation scheduling problem in duty-cycled networks with a single UAV. We propose an adaptive clustering method that takes into account the trajectory and speed of the UAV. The transmission schedule of IoT devices and the UAV departure times are jointly computed so that (1) the UAV flies continuously throughout the shortest path among the CHs to minimize the hovering time and energy consumption, and (2) data are aggregated at each CH right before the UAV arrival, to maximize the data freshness. Intensive simulation shows that the proposed scheme reduces up to 35% of the aggregation delay compared to other benchmarking methods.

Keywords: UAV; data aggregation; scheduling; minimum delay; internet of things

Citation: Nguyen, T.-D.; Pham Van, T.; Le, D.-T.; Choo, H. Adaptive Clustering and Scheduling for UAV-Enabled Data Aggregation. *Electronics* **2024**, *13*, 3322. https://doi.org/10.3390/electronics13163322

Academic Editor: Ping-Feng Pai

Received: 10 July 2024
Revised: 15 August 2024
Accepted: 20 August 2024
Published: 21 August 2024

1. Introduction

Gathering sensory data in a minimal amount of time has been a critical concern for years. Examples of time-sensitive applications are smart monitoring and object tracking, in which sensor devices can be deployed over a large area and the applications need to acquire data in a short time [1]. In reality, while sensors are typically battery-powered [2], they must serve for years. One of the most prominent methods used to prolong the sensor lifetime is duty-cycling [3]. Nevertheless, it lengthens the data-gathering time and degrades the application service quality. In this regard, UAVs potentially help, and research studies on UAVs are receiving increasing interest from both academia and industry. A UAV can serve as a mobile station, a relay node, or an edge [4]. Particularly, for data collection applications, UAV is helpful in various ways. For example, it prolongs the sensor network lifetime and increases channel quality by providing direct short-range communication with sensor devices [5]. It can also fly over an unattended/disconnected area to gather data, which lowers the deployment costs.

This paper investigates the minimum-latency data aggregation scheduling problem in duty-cycled wireless sensor networks with UAVs. Herein, a UAV is responsible for collecting data from the ground sensors. Because the sensor data are highly correlated, data aggregation techniques are often employed to reduce time, energy consumption, and redundancy. The network splits into several clusters, each of which is managed by a cluster head. Sensor nodes in a cluster send their data to the cluster head (CH), and the UAV only visits these CHs to acquire the aggregated data. Several prior works presented different

optimization perspectives for the UAV-enabled data collection problem. authors of [6,7] arranged multiple UAVs to minimize the energy consumption of sensor devices. In [8], the authors emphasized that the age of information should be minimal, but the scheme only facilitated a small number of sensors. Aiming at a similar goal, in [9], the authors assumed that data become obsolete after a certain deadline. A UAV is intended to gather data from as many devices as possible. Quite a few other researches aimed to minimize the UAV-related metrics, e.g., UAV flight time and UAV energy consumption. These works either require UAV to approach every device [6], or divide the network into small-size clusters with direct communication between CH and cluster members (CMs) [10]. This would result in an excessive number of clusters if the network size increases, making the UAV path planning and scheduling more challenging. Furthermore, only optimizing the UAV itinerary might not be efficient, because the data collected at each cluster head can become obsolete after a certain waiting time. To the best of our knowledge, there has been no research on delay-efficient transmission scheduling for sensor networks in conjunction with UAV itinerary scheduling.

Herein, we propose an adaptive clustering and scheduling (ACS) approach that computes the device transmission time tightly based on the UAV itinerary. Each CH is given a deadline to aggregate data from its surroundings. Because the UAV visits a sequence of CHs, the $(k+1)$-th CH has more time to gather data than the k-th one. As such, later-visited clusters have larger deadlines and should contain more CMs. Inside each cluster, device transmissions are pipelined based on their duty cycle to maximize the number of devices it can accommodate within the deadline. Our contributions are briefly stated as follows. Note that, the terms 'sensor' and 'device' are used interchangeably in this paper.

- We formulate a joint UAV-sensor device scheduling problem and prove its NP-hardness. Then a simpler problem is derived in which the UAV can fly continuously without hovering at each CH.
- The proposed scheduling approach precisely computes the transmission schedule among devices and to the UAV. By calculating the times at which the UAV should approach the CHs, we infer the deadline for each CH to acquire data from the CMs.
- From the computed deadlines, we design a collision-free reverse cluster formation and scheduling scheme. Herein, the scheme simultaneously constructs a data aggregation tree in each cluster and assigns transmission times to devices. Note that the last CH with the largest deadline will start to construct the aggregation tree and acquire CMs first, followed by the CH with the second largest deadline, and so forth.

The remainder of this paper is organized as follows. Section 2 reviews recent work related to the problem we are solving. In Section 3, we describe network models and assumptions, followed by the problem formulation in Section 4. The proposed ACS scheme is presented in Section 5. We evaluate the proposed design in Section 6, summarize the contributions of this work, and discuss future directions in Section 7.

2. Related Work

For decades, a mobile sink (or mule) has been used as a solution to collect data from ground sensors [11]. A mobile sink is useful in situations where there are no dedicated and powerful nodes to collect data from the surroundings, or when the network is disconnected. However, the ground mobile sink is affected by complex terrains and obstacles; therefore, they lack flexibility and speed [11]. Consequently, it might take a lot of time to collect data from a network. This is problematic since, nowadays, data freshness is also a critical concern. UAV seems to be a viable replacement for the ground mobile sink. Existing research aimed to shorten the UAV itineraries, since the UAV resources are usually limited. For instance, in [12,13], the authors assumed that the UAV collects individual sensor data following a fixed route. Herein, not all sensors send data to the UAV. In [14], the authors took into account the altitude, trajectory, velocity, and wireless link when minimizing data collection time. The UAV shall hover at a maximum altitude so as to maximize the

transmission range, thereby shortening the trajectory. Concerning the age of information, the authors of [8] also targeted a minimal UAV trajectory and proposed solutions using dynamic programming and genetic algorithms. With the same objective, the authors of [15] modeled the data collection problem as a Markov decision process, in which they used deep reinforcement learning to find an optimal UAV trajectory. In [16], the authors solved the trade-off between the energy consumption of the UAVs and the number of UAVs required to cover an area. Herein, fewer UAVs mean that each UAV has to cover a larger area. As a consequence, the UAVs have to hover at a higher altitude for a longer period, which drains more energy. In general, prior works, that focused on UAV performance involved determining the traveling order of CHs, differing in how the constraints were considered. Several other studies in this area, such as [17–19], have also contributed to this direction.

UAVs can help improve the energy efficiency of WSNs (wireless sensor networks), because traditional data collection solutions for WSNs are used to create energy holes around the sink due to the high traffic going through these nodes [11]. UAVs, instead, can freely move to any location and alleviate the energy hole problem. For example, the authors of [20] let the UAV move close to the target sensors, so sensor devices did not need to transmit at high power. In [21], the authors optimized the transmitting power and transmission reliability of Internet of Things (IoT) devices by clustering and using multiple UAVs, one per cluster. In [10,22], the network is split into clusters of equal sizes to balance the energy consumption of sensors and prolong the network lifetime. In [13], the scheme allows the prioritization of the region of interest and ignores sensors from low-priority regions. In [6], the authors presented a successive convex optimization technique to achieve a sub-optimal solution. Herein, not only is reliable communication with sensors guaranteed, but the energy consumption of all sensors is also minimized. Lee et al. proposed a LEEF algorithm, which utilizes k mobile relays to cover k clustered regions. LEEF strives to equalize the energy consumption of mobile relays [23].

To the best of our knowledge, although there were many efforts conducted on data collection with UAV, all the existing schemes only focus on the performance metrics of the UAV without considering the aggregation schedule inside each cluster. Moreover, the clusters usually have equal size and are organized in a star topology, which requires a long time to collect data as the CMs have to take turns sending to the CH. Finally, the geographical coverage of a CH is limited by its communication range; therefore, to maintain the direct transmissions between CMs and CH, the number of clusters increases when the deployment area scales up. As a result, the UAV itinerary becomes longer and more inefficient. Our proposed scheme, ACS, arranges to aggregate data at the CHs right before the arrival of the UAV. In addition, the UAV speed and CH visiting order will determine the cluster size. We utilize the sleep–awake pattern of sensors to pipeline transmissions through multi-hop routes such that each CH can accommodate a larger number of CMs in a short aggregation time.

3. Network Model

We model the network as a graph $G = (V, E)$, in which V is the set of homogeneous IoT devices (nodes) and E is a set of communication links between the devices. Herein, there exists a link between two nodes $u, v \in V$ if and only if the Euclidean distance between them is smaller than the communication range R. The whole network shares a single communication channel. Two neighbors can communicate with each other, and if a node transmits, all of its neighbors can hear the signal. This leads to collisions if two or more signals simultaneously arrive at the same receiver. The node then cannot receive any of the signals. We use the protocol interference model, which consists of primary and secondary collisions [3].

Sensor devices are temporally synchronized and use TDMA (time division multiple access) to access the channel. Herein, time is divided into slots and each sensor device sends the data at a specific slot. There will be no collision avoidance mechanism required, such as CSMA (carrier sensing multiple access). Instead, it requires a transmission schedule for the

whole network, so that each device can send data without collision. The communication transceivers operate in half-duplex mode, i.e., at a time, they can only either send or receive data. We assume that all devices run in duty cycle mode, in which a device only wakes up for a short time to receive or transmit data, and for the rest of the time, it stays in a sleeping state to conserve energy. Herein, time consists of equal working periods, and each working period is further split into constant T slots with indices in $\{0, 1, \ldots, T-1\}$. Each device randomly chooses an active slot $\tau \in \{0, 1, \ldots, T-1\}$. The device periodically wakes up at its active slot to wait for possible incoming transmission and sleeps otherwise. If a device wants to send data to its neighbor, it must wake up and transmit data at the active slot of the receiver. Figure 1 illustrates a network containing three nodes $\{a, b, c\}$, in which node c is active at slot 0 and $T = 3$. Nodes a and b must send data to node c in two different working periods, at the active slot 0 of node c. A time slot is sufficient to transmit a packet.

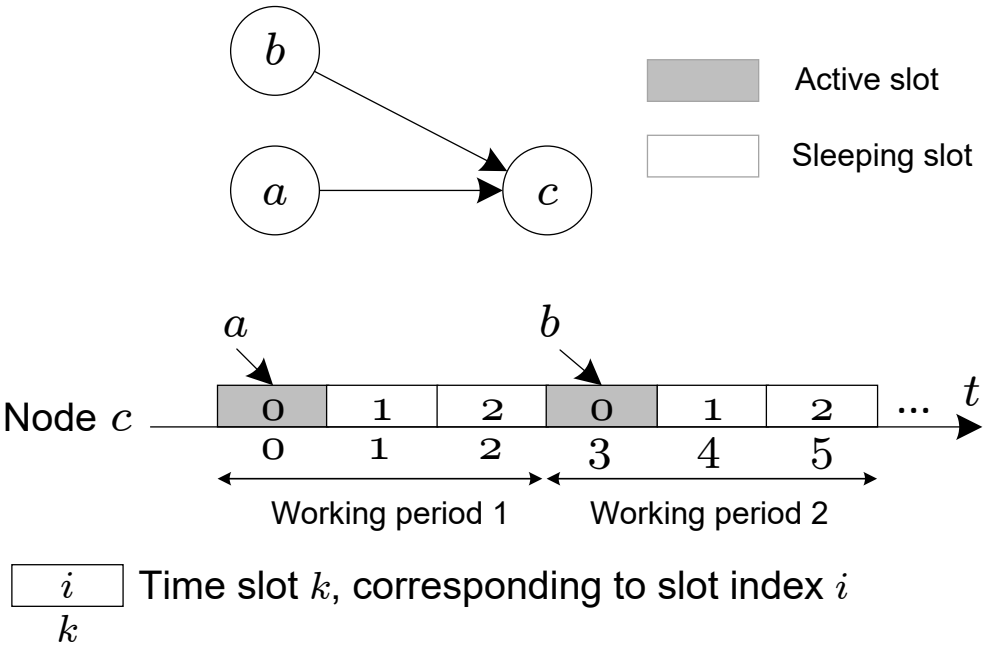


Figure 1. An example of a duty-cycled network. Node c is active at slot 0 and each working period consists of 3 slots. Two senders a and b must take turns sending data to node c in two different working periods.

Sensor data sent by the devices can be aggregated perfectly [3] so that each node only sends data once. This means that, regardless of the number of received data packets, the intermediate node will aggregate the received data with its data, to form a single packet toward the upstream device. We also assume that the UAV is navigated and positioned so that its wireless links to CHs are sufficiently reliable to guarantee transmission success.

4. Problem Formulation

Table 1 lists the acronyms and notations we use in this paper.

The UAV is assumed to rest at a docking station and remains charged if it is not on duty. On a trip to gather data, it visits a set of CHs, one after another. In each cluster, CH is responsible for aggregating data from other CMs, through multi-hop paths. An intermediate node combines all received data and its own data, then outputs one outgoing transmission. Examples of this aggregation function are MIN, MAX, and AVERAGE. For devices, an aggregation schedule must specify the time slot and the device that sends data to which receiver. In addition, the schedule must incorporate all devices. A valid schedule should induce no collision between transmissions occurring at the same time. For UAVs, the schedule should indicate the departure time, given the known trajectory and speed.

Within each cluster, the routing structure forms a tree topology rooted at the CH. Hereafter, we use the terms sender/receiver and child/parent interchangeably.

Table 1. Notation.

Notation	Explanation
CH	cluster head
CM	cluster member
$a(u)$	active slot of node u
T	number of slots in a working period
R	communication range of devices
v	UAV speed
$d(u,v)$	Euclidean distance between two nodes $u, v \in V$
$N(u)$	neighbor set of node u
$N_H(u)$	neighbors of node u that are also in the set H
$C(u)$	children set of node u
c_0	the docking station where the UAV starts from, c_0 does not aggregate data from any device.
c_i	the i-th CH in which order the UAV will visit ($i \in [1,k]$)
$\mathcal{C}$	the set of CHs, $\mathcal{C} = \{c_i\}_{i\in[1,k]}$
V_i	set of nodes in the i-th cluster, excluding the CH c_i ($i \in [1,k]$)
A_i	the arrival time of the UAV at the i-th cluster ($i \in [1,k]$)
D_i	departure time at the i-th cluster ($i \in [1,k]$)
$\mathcal{T}_i$	the tree rooted at c_i. $\mathcal{T}_i$ is a data structure of the sender–receiver relationship among CMs in the i-th cluster ($i \in [1,k]$)
$tx(u)$	transmission time slot of node u
S_i	set of nodes in V transmitting at time slot i
$l(u,v,T)$	link delay from node u to node v given working period length T

The aggregation time, or aggregation delay, is the time elapsed from when the UAV departs from the docking station, until when it leaves the last CH in its itinerary after successfully retrieving data. Let k be the number of clusters and $\mathcal{C} = \{c_i\}_{i\in[1,k]}$ is the set of CHs, which are known in advance. For example, several nodes with more resources can be deployed in the field as CHs. The CH locations may depend on the terrain. Finding an optimal number k as well as the optimal positions of the CHs are postponed to future research. Because the whole network is partitioned into k clusters, we have:

$$\bigcup_{i\in[1,k]} V_i = V \backslash \mathcal{C}. \tag{1}$$

Remark 1. *Given the locations of the CHs, we can compute the shortest flying path for the UAV. For simplicity, we assume that the set $\mathcal{C}$ is ordered and represents the shortest trajectory. Nevertheless, the size and members V_i of each cluster depend on the scheduling scheme.*

Let $\zeta_i, i \in [0, k-1]$ be the traveling time from CH c_i to CH c_{i+1} and α_i, $i \in [0,k]$ be the waiting time at the c_i (Figure 2). Both ζ_i and α_i are measured in the number of time slots. Let v be the speed of the UAV (m/ time slot), we have the following:

$$\zeta_i = \lceil \frac{d(c_i, c_{i+1})}{\mathrm{v}} \rceil, \quad \forall 0 \le i \le k-1. \tag{2}$$

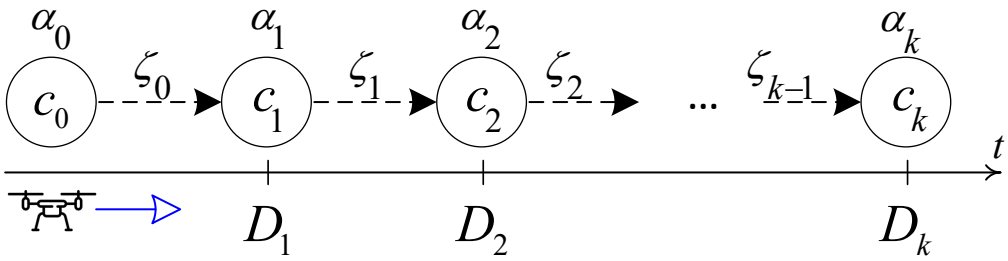


Figure 2. An illustration of the hovering and flying times from the docking station to the CHs $\{c_i\}_{i\in[1,k]}$. Dashed arrows indicate the sequence of visits. The blue arrow indicates the moving direction of the UAV. The α_i is the hovering time at c_i, and ζ_i is the flying time from c_i to c_{i+1}, $i \in [0,k]$.

The hovering time $\alpha_i > 0, i \in [1,k]$ if and only if CH c_i has yet to finish data collection upon the UAV arrival. We can compute the UAV arrival time A_i at CH c_i as follows:

$$A_i = \sum_{j=0}^{i-1}(\alpha_j + \zeta_j), \quad \forall 1 \le i \le k. \tag{3}$$

The departure time D_i of the UAV at CH c_i, i.e., the data aggregation time at the i-th cluster, is as follows:

$$D_i = \alpha_i + A_i = \alpha_i + \sum_{j=0}^{i-1}(\alpha_j + \zeta_j), \quad \forall i \in [1,k]. \tag{4}$$

Note that $\{\zeta_i\}_{i\in[0,k-1]}$ are constant values, and D_k is the aggregation time of the whole network. To minimize D_k, one needs to minimize the total waiting time $\sum_{j=0}^{k}\alpha_j$. Herein, the total waiting time comprises the hovering time $\sum_{j=1}^{k}\alpha_j$ and the waiting time α_0 at the base station. The selection of $\{\alpha_i\}_{i\in[0,k]}$ must ensure that all devices participate in the data aggregation.

Remark 2. *D_k is the aggregation time of the whole network.*

Let $\mathfrak{S} = \{S_i\}_{i\in[1,D_k]}$ be an arbitrary valid transmission schedule, in which S_i is the set of devices transmitting data at time slot i: $S_i = \{u \in V\backslash\mathcal{C} | tx(u) = i\}$. The following conditions must hold:

$$\bigcup_{1\le i\le D_k} S_i = V\backslash\mathcal{C}, \tag{5}$$

and

$$S_i \cap S_j = \emptyset, \quad \forall 1 \le i \ne j \le D_k. \tag{6}$$

Herein, (5) means that all the devices are assigned transmission time slots, and (6) ensures that each node only sends data once to its receiver. In addition, all the transmissions occurring at the same time must be collision-free, $\forall i \in [1, D_k]$:

$$u \notin N(p(v)) \text{ and } v \notin N(p(u)), \quad \forall u, v \in S_i, u \ne v. \tag{7}$$

Within a cluster, the transmission time of each node must be smaller than the aggregation time:

$$tx(u) \le D_i, \forall u \in V_i. \tag{8}$$

Furthermore, since a parent node may receive data from several children, the transmission time slot of a child must be smaller than that of its parent.

Remark 3. *Due to the aggregation constraint, the transmission time slot of a parent must be later than that of its children, i.e.,*

$$tx(u) > tx(v), \quad \forall v \in C(u). \tag{9}$$

An aggregation schedule can be identified by the routing structure, i.e., $\{\mathcal{T}_i\}_{i\in[1,k]}$, which points out a receiver for each device, and the transmission time, i.e., $\{S_j\}_{j\in[1,D_k]}$, which enforces when the devices should send data. We can formally formulate the optimization problem as follows.

$$\begin{aligned} \min \quad & D_k | \exists \{\mathcal{T}_i\}_{i\in[1,k]}, \{S_i\}_{i\in[1,D_k]} \\ \text{subject to} \quad & (4)\text{–}(9). \end{aligned} \tag{10}$$

Theorem 1. *The problem* (10) *is NP-hard.*

Proof. Let us consider a reduced problem of (10) when $k = 1$. This problem becomes the conventional minimum-time aggregation scheduling problem in wireless sensor networks, and is NP-hard [24]. Therefore, problem (10) is NP-hard. □

5. Adaptive Clustering and Scheduling Scheme

This section presents a heuristic scheme to solve problem (10). Recall that there are two types of scheduling for WSNs. We summarize the main features as follows.

- **Bottom-up scheduling.** This type requires to construction of an aggregation tree rooted at the CH in advance. Next, the scheduling algorithm identifies the transmission time slot for each tree edge, from the leaves up to the root. The scheduling algorithm iteratively finds transmissions for time slots $1, 2, \ldots, D_k$.
- **Top-down scheduling.** This type does not require having a tree beforehand. The tree is grown during the scheduling process. Unlike the bottom-up approach, we can provide a deadline to collect data at the root node. Depending on the given deadline, a part or the whole network can send data to the root [25]. The scheduling algorithm iteratively finds transmissions for time slots $D_k, D_k - 1, \ldots, 2, 1$, in the reverse order of the bottom-up scheduling approach.

Regarding the bottom-up approach, the process begins with (step 1) splitting the network into clusters, followed by (step 2) constructing the aggregation tree in each cluster, and then (step 3) performing a scheduling algorithm. However, how to optimally cluster the network is hard, as it is impossible to know the aggregation time in each cluster without running step 2 and step 3. This may require numerous trials and errors to achieve a sufficient solution.

On the other hand, top-down scheduling allows for the simultaneous construction of the aggregation tree and scheduling, and the tree size (and, hence, the cluster size) can be adjusted together with the given aggregation deadline. Figure 3a illustrates a case where three CHs are present, and the UAV leaves the CHs at D_1, D_2, and D_3. In Figure 3b, the clusters may have different sizes because the aggregation deadline for each is different. While all the clusters consist of members being scheduled until D_1, only CH_2 and CH_3 continue to aggregate data during $D_1 < t \leq D_2$, and then only CH_3 contains transmissions scheduled during $D_2 < t \leq D_3$. Generally, when the UAV flies to the CH_1, the sensor devices can transmit data toward any CH. After the CH_1 delivers data to the UAV, the remaining CHs, i.e., $\{CH_2, CH_3, .., CH_k\}$, continue to gather data. Since the UAV may arrive at the i-th CH earlier than the time c_i finishes collecting data from its cluster, it may have to wait for the duration $\alpha_i \geq 0$ before departure. Based on D_k, the data aggregation deadline at the i-th cluster can be derived from (4) as follows:

$$D_i = D_k - \sum_{i+1}^{k} \alpha_j - \sum_{i+1}^{k-1} \zeta_j. \tag{11}$$

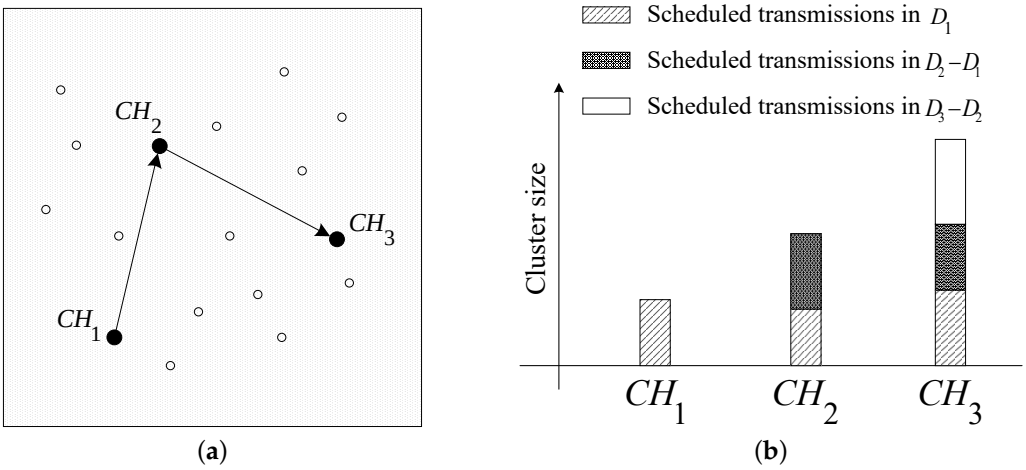
The following theorem proves that we can always simplify problem (10) by adding a constraint on the waiting time at each CH (or hovering time) as follows:

Theorem 2. *For any valid aggregation schedule* $\mathfrak{S}$ *with* $\mathcal{D} = \{D_i\}_{i\in[1,k]}$*, there always exists a schedule* $\mathfrak{S}'$ *with* $\mathcal{D}' = \{D'_i\}_{i\in[1,k]}$ *such that* $\sum_{j=1}^{k} \alpha'_j = 0$ *and* $D'_k = D_k$*.*

Proof. We can identify $\{D'_i\}_{i\in[1,k]}$ as follows:

$$D'_i = D_k - \sum_{j=i}^{k-1} \zeta_j, \quad \forall i \in [1, k-1]. \tag{12}$$

From (12) and (4), we have $D'_i \geq D_i, \forall i$ because $\alpha_i > 0, \forall i \in [1, k]$. Since each CH c_i can aggregate data from its cluster within D_i time slots, it can also aggregate data within $D'_i \geq D_i$ time slots. The theorem is proved. □



(**a**) (**b**)

Figure 3. Sample UAV trajectory to three CHs. The arrival times at each CH are $D_1 < D_2 < D_3$. With different time allowances to collect data, CH_1, CH_2, and CH_3 may have different cluster sizes. (**a**) UAV trajectory; (**b**) CHs and the corresponding sizes.

Using Theorem 2, we can simplify the problem in (10) to the following problem: Find min D_k and a scheduling algorithm $\mathcal{S}$ such that when giving the deadlines D_i to the CHs $\{c_i\}_{i\in[1,k]}$, all devices are scheduled to send data to the CHs. The scheduling algorithm $\mathcal{S}$ must satisfy Equations (5)–(9). Herein, D_i is calculated by the following:

$$D_i = D_k - \sum_{j=i}^{k-1} \zeta_j, \quad \forall i \in [1, k-1], \tag{13}$$

or, equivalently:

$$D_i = \alpha_0 + \sum_{j=0}^{i-1} \zeta_j, \quad \forall i \in [1, k], \tag{14}$$

which is simpler than (4), as the variables $\{\alpha_i\}_{i\in[1,k]}$ are eliminated.

In reality, the UAV might not be able to fly continuously, while ensuring reliable communication with the selected CHs. However, we can consider v as the average speed of the UAV, taking into account the hovering time and acceleration/deceleration time.

5.1. Overall Approach

Equations (13) and (14) suggest that the aggregation deadlines at all the clusters can be derived from the aggregation deadline at the last CH D_k, or from the waiting time α_0 at

the docking station c_0 as the hovering times at all the CHs are zero ($\{\alpha_i\}_{i\in[1,k]} = 0$). In other words, once either α_0 or D_k is selected, then $\{D_i\}_{i\in[1,k]}$ is deterministic. Given a scheduling algorithm $\mathcal{S}$, Algorithm 1 shows a naive method to determine the minimum value of α_0. The first α_0 is returned if the derived deadlines $\{D_i\}_{i\in[1,k]}$ given to the CHs are sufficient to cover the entire network. However, this approach requires repeating the same scheduling procedure (lines 3–4) multiple times, making it costly.

Algorithm 1 Naive method to find α_0.

Input: $G = (V, E)$, set of CHs $\mathcal{C}$

Output: Minimum α_0

1: **while** True **do**
2: Compute $\{D_i\}_{i\in[1,k]}$ as in (14)
3: **if** $\mathcal{S}$ already covered all nodes within $\{D_i\}_{i\in[1,k]}$ **then**
4: **break**
5: **else**
6: $\alpha_0 \leftarrow \alpha_0 + 1$
7: **end if**
8: **end while**

Instead, this paper proposes a novel adaptive top-down data aggregation scheduling approach, namely ACS, which grows the aggregation trees from the roots (i.e., the CHs), and at the same time computes the transmission schedule. Herein, ACS starts at the last transmission time slot, i.e., D_k, and goes backward to find the senders in each time slot. Assuming that $\mathcal{R}$ denotes the reverse scheduling algorithm, Algorithm 2 illustrates an abstract procedure of ACS.

Algorithm 2 One-shot reverse method to find D_k.

Input: $G = (V, E)$, set of CHs $\mathcal{C}$

Output: The aggregation time.

1: Assign some big value to D_k
2: Compute $\{D_i\}_{i\in[1,k-1]}$ as in (13)
3: Run the scheduling algorithm $\mathcal{R}$
4: $t_{min} = \min_{u\in V\setminus\mathcal{C}} \; tx(u)$
5: **for** $u \in V\setminus\mathcal{C}$ **do**
6: $tx(u) \leftarrow tx(u) - T\lfloor \frac{t_{\min}}{T} \rfloor$
7: **end for**
8: **return** $D_k - T\lfloor \frac{t_{\min}}{T} \rfloor$

At first, ACS initializes a sufficiently large deadline D_k for the last cluster (line 1), and computes other deadlines for the remaining CHs (line 2). This is to ensure that after running the scheduling algorithm $\mathcal{R}$ (line 3), all the transmission time slots of devices are positive numbers. Next, we take t_{min} as the smallest time slot used to schedule (line 4). Because the transmissions are scheduled in a reverse direction, there will be no transmission scheduled for time slots $1, 2, 3, ..., t_{\min} - 1$ (Figure 4). As a result, we can shift all devices' assigned transmission time slots backward by $T\lfloor \frac{t_{\min}}{T} \rfloor$ time slots, i.e., by $\lfloor \frac{t_{\min}}{T} \rfloor$ working periods to the left (lines 5–6). The aggregation delay is, thus, $D_k - T\lfloor \frac{t_{\min}}{T} \rfloor$.

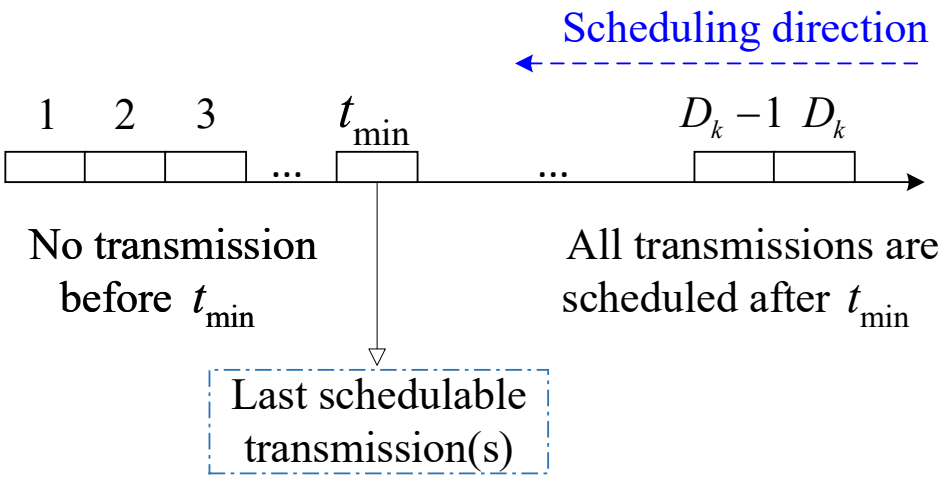


Figure 4. Reverse scheduling approach with a large D_k. The algorithm finds the transmissions in each time slot, from the latest to the earliest. The last schedulable transmissions are in time slot $t_{\min}$. There is no transmission in time slots $[1, t_{\min}-1]$. All the transmissions are scheduled in $[t_{\min}, D_k]$.

Figure 4 shows why we can shift the schedule be a maximum of $\lfloor \frac{t_{\min}}{T} \rfloor$ working periods to the left. Because the algorithm computes the schedule in reverse order, all the transmissions are assigned time slots in the range $[t_{\min}, D_k]$. However, it is not possible to shift the schedule by $t_{\min}$ time slots because the new schedule may break the duty cycle settings of the receiver. For example, if $t_{\min} \mod T \neq 0$, and a transmission from u to v is scheduled at $t_{\min}$. It means that $a(v) = t_{\min} \mod T \neq 0$. If the algorithm shifts $t_{\min}$ time slots, then u will be scheduled to send data to v at time slot 0, which is not at the active slot of v. The maximum number of slots that we can shift is, therefore, $T\lfloor \frac{t_{\min}}{T} \rfloor$.

Remark 4. *After initializing with some large deadline D_k and finishing with the earliest transmission at $t_{min} > 0$, the transmission schedule is shifted backward by $\lfloor \frac{t_{\min}}{T} \rfloor$ working periods. The aggregation delay is then $D_k - T\lfloor \frac{t_{\min}}{T} \rfloor$ time slots.*

In the next subsection, we will provide a detailed design of the scheduling algorithm $\mathcal{R}$.

5.2. Adaptive Clustering and Scheduling

The proposed ACS scheme is presented in Algorithm 3. For the ease of scheduling implementation, we assign transmission time slot $D_i + 1$ to CH c_i for all $i \in [1, k]$ (lines 1–2). This can be seen as the time c_i transmits data to the UAV. Typically, the algorithm iterates the backward time slot by the time slot, starting from the deadline D_k at the last CH c_k. Herein, the set of candidate receivers P is determined first, followed by finding the senders for them. The set of scheduled devices is denoted by SCH and is initially empty (line 4). ACS stops when all the devices are scheduled, i.e., $|SCH| = |V \backslash \mathcal{C}|$. Lines 5–15 present the process in which ACS searches for eligible transmitting nodes at time slot t. Herein, P is the set of parent candidates at the current time slot t, which is initially empty. A parent candidate $u \in P$ (i.e., u can be a receiver) at time slot t is any node that satisfies the conditions below:

1. $tx(u) > t$ (line 8): candidate receiver u is already scheduled. By checking this condition, ACS makes sure that a CH only starts acquiring CMs to grow its aggregation tree at a proper time. For example, when Algorithm 3 starts at $t = D_k$, only c_k grows its aggregation tree. When the time t decreases down to $t = D_{k-1}$, then c_{k-1} begins constructing its own aggregation tree, and so on.
2. $a(u) = t \mod T$ (line 9): candidate receiver u is active at the current time slot t; otherwise, it cannot receive data.
3. $N(u) \backslash SCH \neq \varnothing$ (line 9): candidate receiver u has at least one unscheduled neighbor. Otherwise, no node can be scheduled to send data to u.

Algorithm 3 ACS scheduling scheme.

Input: $G = (V, E)$, set of CHs $\mathcal{C}$, aggregation delay det at each cluster $\{D_i\}_{i\in[1,k]}$
Output: The transmission schedule $tx(u), \forall u \in V \backslash \mathcal{C}$

1: **for** each $i \in [1, k]$ **do**
2: $\quad tx(c_i) \leftarrow D_i + 1$
3: **end for**
4: $t = D_k$
5: $SCH \leftarrow \varnothing$
6: **while** $|SCH| < |V \backslash \mathcal{C}|$ **do**
7: $\quad P \leftarrow \varnothing$
8: $\quad$ **for** $u \in V \big| tx(u) > t$ **do**
9: $\quad\quad$ **if** $a(u) = t \mod T$ **and** $N(u) \backslash SCH \neq \varnothing$ **then**
10: $\quad\quad\quad P \leftarrow P \cup \{u\}$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad C \leftarrow \{v \in N(P) \big| tx(v) = None\}$
14: $\quad$ Find a collision-free matching between P and C
15: $\quad C_m \leftarrow$ set of senders in the matched pairs
16: $\quad tx(u) \leftarrow t \quad , \forall u \in C_m$
17: $\quad SCH \leftarrow SCH \cup C_m$
18: $\quad t \leftarrow t - 1$
19: **end while**

These nodes in P are the receiver candidates. The set of candidate senders C consists of nodes that are neighbors of at least one node in P and are not scheduled (line 13). In order to identify the sender–receiver pairs that can transmit data simultaneously without collision, ACS performs matching between P and C. The detailed matching algorithm is presented in Algorithm 4. After matching, selected senders will be assigned the transmission time t (line 16), and are added to the SCH set. Then, the algorithm proceeds to the next time step: $t \leftarrow t - 1$ (line 18).

Finding a matching between the sender set C and the receiver set P can be performed in various ways, depending on how we prioritize the nodes. For example, in our previous work [26], we used the node degree to sort the set P. This work nevertheless targets the low duty-cycled WSNs, for which we utilize two well-known metrics, link delay and number of neighbors, as follows.

Definition 1. *Link delay from a sender u to a receiver v is as follows:*

$$l(u, v, T) = \begin{cases} a(v) - a(u), & \text{if } a(v) > a(u) \\ a(v) - a(u) + T, & \text{otherwise.} \end{cases}$$

Definition 2. *The neighbors in a set H of nodes u are as follows:*

$$N_H(u) = \{v \in H | v \in N(u)\}. \tag{15}$$

Herein, node u, after finishing data gathering from $C(u)$, must wait for a certain time until its intended receiver v becomes active. The link delay basically means the minimum of such a waiting time.

Algorithm 4 Sender–receiver matching.

Input: Candidate receivers P, candidate senders C

Output: The set of scheduled nodes C_m

1: Sort P in a non-decreasing number of the number of its neighbors in C
2: $C_m \leftarrow \varnothing$
3: **while** $P \neq \varnothing$ **and** $C \neq \varnothing$ **do**
4: $v \leftarrow$ first node in P
5: $u \leftarrow \underset{w \in N_C(v)}{\arg\min} \; l(w, v, T)$
6: $u.parent \leftarrow v$
7: $P \leftarrow P \backslash N(u)$ {Due to collision with the transmission from u}
8: $C \leftarrow C \backslash N(v)$
9: $C_m \leftarrow C_m \cup \{u\}$
10: **end while**
11: **return** C_m

Algorithm 4 seeks a collision-free set of transmissions that can happen at the current time slot t. The receivers are in P and senders are in C. Items in P are sorted such that any preceding item has no more neighbors in C than its succeeding ones, i.e., a node with fewer neighbors in C has a higher priority (line 1). After that, the algorithm picks up a node in C with the smallest link delay to v as the sender (lines 4–5). Next, the algorithm removes all the receiver candidates that overhear the transmission from the selected sender u (line 6) from set P and removes all sender candidates whose transmissions can cause a collision at the selected receiver v (line 7). The **while** loop continues until there is no node left in either P or C.

5.3. Complexity Analysis

In this section, we cover the time complexity analysis of the ACS scheme. We omit the analysis of Algorithm 1, as this algorithm is not used. Below, we will examine the algorithm complexity of Algorithms 2–4.

Algorithm 4 first sorts the candidate receiver set P, which can be conducted in $O(|V| \log |V|)$ time. Line 3 loops $O(|V|)$ times. In the loop, line 5, line 7, and line 8 each take no more than $O(|V|)$. Therefore, the time complexity of Algorithm 4 is $O(|V|^2 \log |V|)$.

In Algorithm 3, lines 1–5 require no more than $O(|V|)$ to complete. Line 6 loops, at most, $|V|$ times. In each loop, the algorithm checks each node in the network vertex set (line 8); for each node u, it will examine the active slot and the aggregation status of the neighbors of node u (line 9). Checking the aggregation status of node u depends on the network density, i.e., the number of neighbors. In the worst case, this operation could require at most $O(|V|)$. Then, lines 13, 16, 17, *and* 18 require $O(1)$. Line 14 is Algorithm 4, which requires, at most, $O(|V|^2 \log |V|)$. Hence, the time complexity of Algorithm 3 is $O(|V|^3 \log |V|)$. It is trivial to see that the main computation part of Algorithm 2 is line 3, which is Algorithm 3. Therefore, the time complexity of the ACS scheme is $O(|V|^3 \log |V|)$.

5.4. Step by Step Example

Figure 5 illustrates a complete run of the ACS on a network topology shown in Figure 5a. Nodes $c_2 = A$ and $c_1 = B$ are two CHs, and the algorithm starts with $D_2 = 25$ and $D_1 = 17$ (time slots). A working period contains $T = 5$ slots.

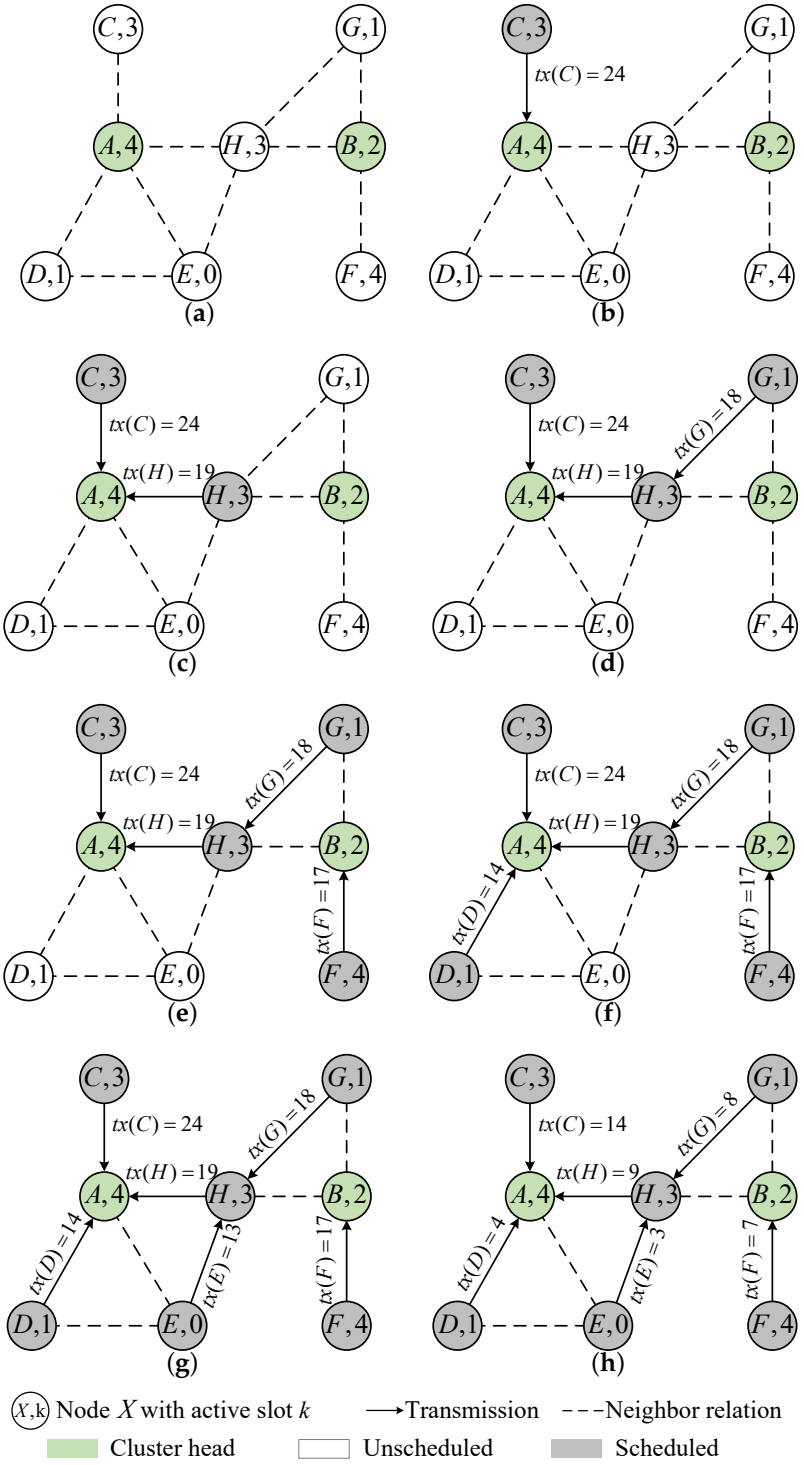


Figure 5. Performing ACS on a sample topology with two CHs $c_1 = B$, $c_2 = A$. Working period length $T = 5$, and the aggregation deadlines $D_1 = 17$, $D_2 = 25$. (**a**) Original topology; (**b**) $t = 24$; (**c**) $t = 19$; (**d**) $t = 18$; (**e**) $t = 17$; (**f**) $t = 14$; (**g**) $t = 13$; (**h**) final schedule.

Although the run starts at $t = 25$, no node is active at slot 0; hence, the algorithm proceeds to time slot $t = 24$. The set of candidate receivers is $P = \{A\}$, and the set of candidate senders is $\mathbf{C} = \{C, D, E, H\}$. Two candidate senders, C and H, have the smallest link delay to A, i.e., $d(C, A, 5) = d(H, A, 5) = 1$. The algorithm randomly chooses one among the two. Here, we assume that node C is selected; hence, $tx(C) = 24$ (Figure 5b).

Next, from time slot $t = 24$, down to time slot $t = 20$, there is no active candidate receiver. Note that, although node C is active at time slot $t = 23$, it does not have any neighbor to be a candidate sender. Then, at $t = 19$, node A becomes active again, and the set of candidate senders is $\mathbf{C} = \{H, D, E\}$. Now H is selected because it has the smallest link delay to A. The transmission time of H to A is $tx(H) = 19$ (Figure 5c). Similarly, when $t = 18$, ACS schedules node G to transmit data to H, so $tx(G) = 18$ (Figure 5d).

In the next time step, $t = 17$, because $D_1 = 17$ and node B is also active at $t = 17$, B is eligible to acquire a sender. The only unscheduled neighbor of B, i.e., node F, is scheduled to send to B at $tx(F) = 17$ (Figure 5e). The algorithm continues with time slot $t = 14$ and $t = 13$ (Figures 5f,g). The final aggregation schedule is obtained in Figure 5h, by subtracting the transmission times of all devices' $\lfloor \frac{13}{5} \rfloor = 2$ working periods, i.e., 10 time slots.

6. Performance Evaluation

6.1. Simulation Settings

We conducted a simulation using Python. The system settings are presented in Table 2. Devices are randomly spread over a 500×500 (m) square area. The transmission range of devices is 50 (m) and we assume that the docking station is located at position $(0, 0)$ as depicted in Figure 6. The UAV's speed is constant during its flight. The network side length L and the transmission range could be varied together without affecting the results. For example, we have the following:

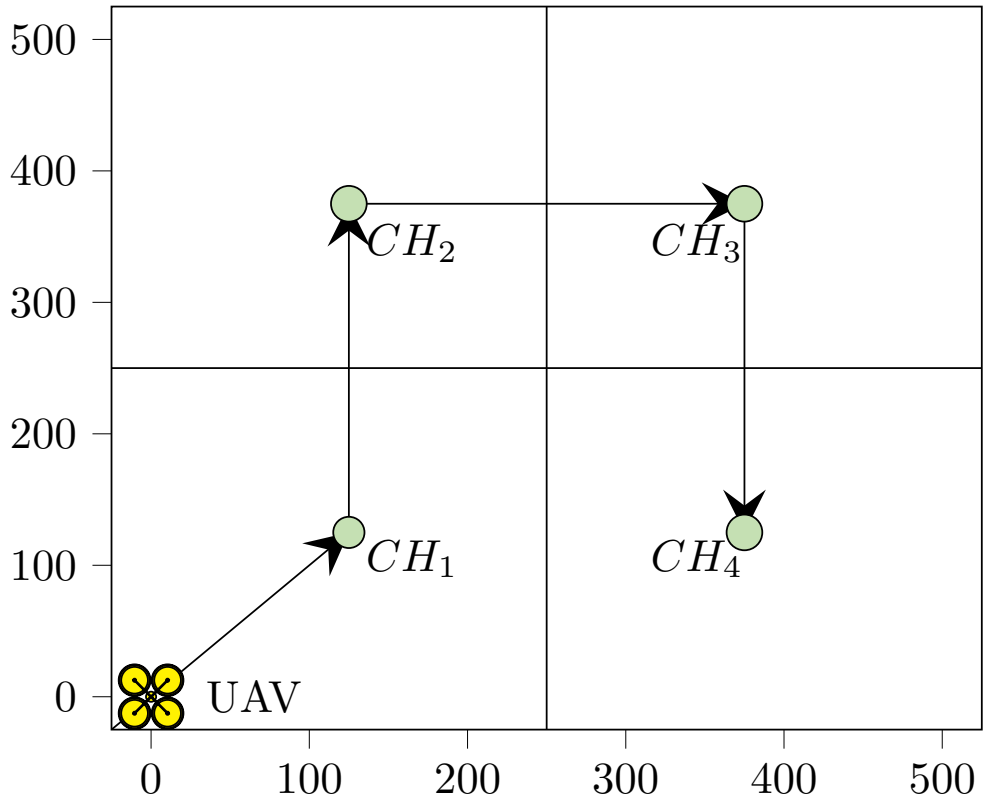


Figure 6. The network is deliberately divided into four equal partitions. Each partition has one CH near the center point. The arrows point out the arrival sequence.

Table 2. Simulation settings.

Parameter	Value
Network side length L	500 (m)
UAV docking station	$(0, 0)$
IoT device transmission range	50 (m)
Number of devices	500–900
Time slot duration	100 (ms)
UAV speed	0.5–2.5 (m/time slot)

The network is deliberately divided into four equal regions, as in Figure 6, and the UAV starts from the bottom left corner. Since the locations of four CHs are known, obtaining the shortest trajectory is simple. The arrows indicate the order in which the UAV will visit the CHs. For each simulation scenario, we randomly generate 30 network topologies and achieve averaged results.

6.2. Baseline Schemes

The proposed ACS scheme is compared with our preliminary solution, namely the incremental clustering and scheduling (ICS) scheme, and a naive scheduling scheme (NS) [26]. The ICS scheme shares the flexible clustering principle with ACS. Herein, the CHs and the UAV trajectory are, respectively, identical. However, when selecting a sender for each receiver in the parent candidate set (line 4 in Algorithm 4), ICS randomly chooses a sender among the unscheduled neighbors.

For the NS scheme, the devices are grouped into four clusters based on their geographical locations. All devices located inside the bottom left quarter belong to Cluster 1, all devices in the top left quarter belong to Cluster 2, and so on (Figure 6). Each CH (at the center of each quarter) has to aggregate data from the corresponding quarter where it resides. The UAV trajectory is also the same as for ACS and ICS schemes. We again apply Algorithm 3 per cluster, individually, by setting $k = 1$. Because the four clusters have similar sizes, aggregation times at each CH should be close.

6.3. Impact of Network Size

Figure 7 presents aggregation delay when UAV moves at the 2 m/time slot. The network side length L is set to 500 m and we vary the number of deployed devices. When T is relatively small, ACS and ICS have similar performance, and both are around 7.5% better than the NS scheme. However, when T increases, ACS is significantly better. For example, for $T = 50$, aggregation delay values by ACS are about 7% and 32% smaller than those of ICS and NS, respectively. For $T = 100$, the differences are larger, i.e., ACS values are 29% and 42% smaller in delay compared to those in ICS and NS. This is understandable because the two latter algorithms ignore the active sleep patterns of sensor devices, which is not negligible in duty-cycled networks.

When N increases from 500 to 900 nodes, the aggregation delays of the three schemes do not show notable increments. This is because of the relatively sparse density of device deployment, which results in a sufficiently large number of unused time slots in low-duty-cycled settings. When the number of devices increases, the added nodes can utilize the unused time slots to transmit data. Let ρ be the network density, i.e., the average number of nodes in a radius-R disk [25]. For instance, when $N = 500$, $R = 50$, $L = 500$, we have $\rho = \frac{\pi R^2 N}{L^2} = \frac{\pi \times 50^2 \times 500}{500^2} \approx 15.7$ (nodes). As for $N = 900$, the average number of neighbors each node has is 28. Figure 8 shows an example of when several nodes are added to a network. In Figure 8a, the network has two nodes, $\{a, d\}$, with active slots, $\{0, 30\}$, respectively. The working period length here is assumed to be any number larger than 30. Node a transmits data to node d at time slot 30. In Figure 8b, node b with active slot 10 is added to the network, and b is a neighbor of both a and d. Herein, the scheduling algorithm can arrange the transmission from a to b at time slot 10, and the transmission

from b to d at time slot 30. Similarly, we can add another node c to the network and adjust the schedule so that the aggregation time is unchanged, i.e., 30 time slots in this example.

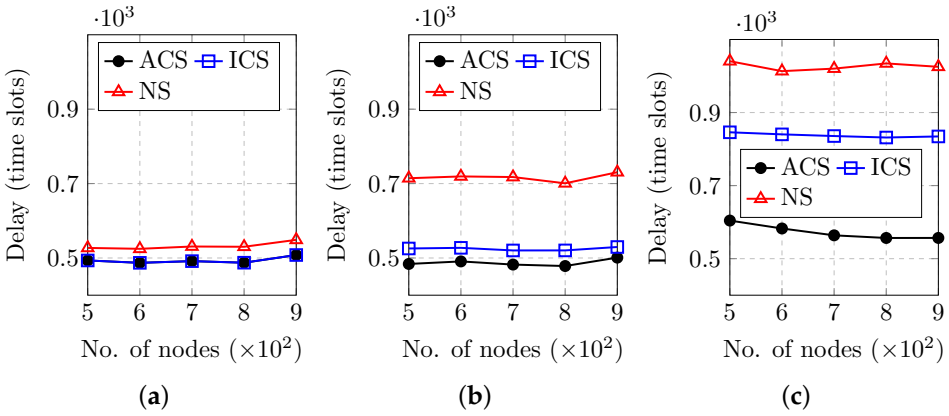


Figure 7. Data aggregation delay with $L = 500$ m and $v = 2$ m per time slot. (**a**) $T = 20$ slots; (**b**) $T = 50$ slots; (**c**) $T = 100$ slots.

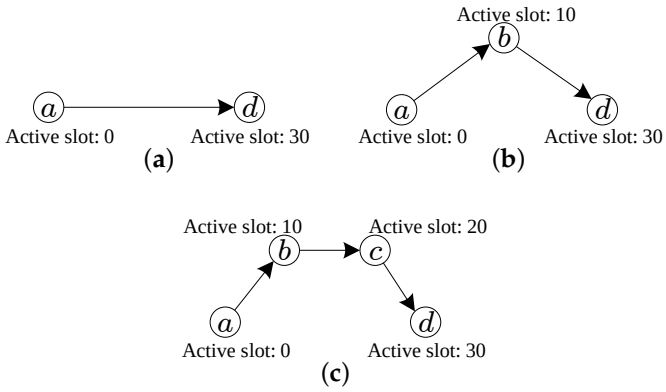


Figure 8. Networks of different sizes but with the same aggregation delay: 30 time slots. (**a**) Two nodes; (**b**) three nodes; (**c**) four nodes.

6.4. Impact of T

To evaluate the impact of T, we set $L = 500$ m, $v = 2$ m/time slot, and vary the working period length T in the range $[10, 200]$ while keeping the number of nodes N constant as in Figure 9. The aggregation delay is computed in a number of working periods.

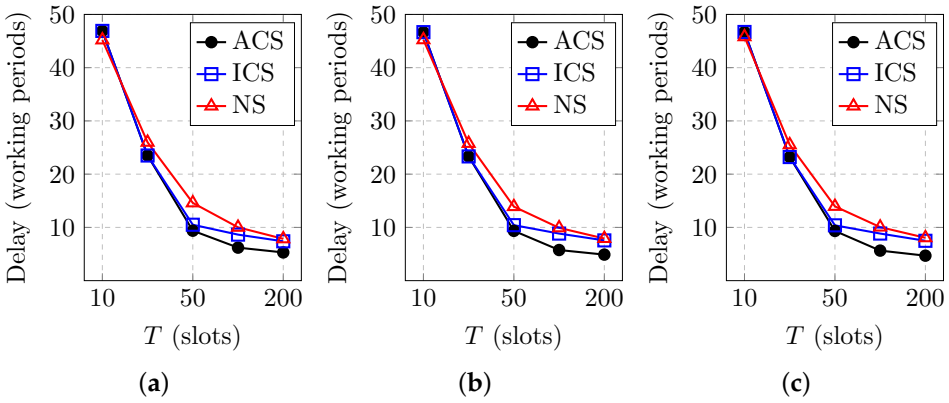


Figure 9. Data aggregation delay with $L = 500$ m, $v = 2$ m per time slot. (**a**) $N = 500$; (**b**) $N = 700$; (**c**) $N = 900$.

When T increases, the aggregation delay in working periods decreases. The reason is that a large T allows more transmissions to occur in a working period. With a small T, aggregation delays of ACS, ICS, and NS are similar. The ACS scheme has notable improvement in low-duty-cycled networks. When $T = 100$, the aggregation delay values of ACS are 35% and 42% smaller than those of ICS and NS, respectively.

6.5. Discussion

ICS and ACS differ in the way they construct the aggregation tree at each cluster. For ACS, the algorithm takes an extra step (line 5 of Algorithm 4), which requires $O(|V|)$ time complexity in the worst case (when the network forms a completed graph). For a sparse network, the required time depends linearly on the number of neighbors of a node, which is basically the network density. Below, we further present observations on the cluster distribution, UAV speed, and departure time.

Distribution of nodes in clusters. Figure 10 depicts the final cluster formation of ICS and ACS schemes. Herein, nodes belonging to the same cluster have the same shape and color. The first cluster has the smallest number of members, and the cluster size increases subsequently. Since ICS builds the trees based on the neighboring relation, the trees span relatively concentrated regions. However, for ACS, the primary concern is the sleep latency between the sender and receiver. Therefore, members of a big cluster can span across multiple regions. For example, Cluster 4 and Cluster 3 consist of nodes in all four regions, while Cluster 2 spans across regions 1, 2, and 3.

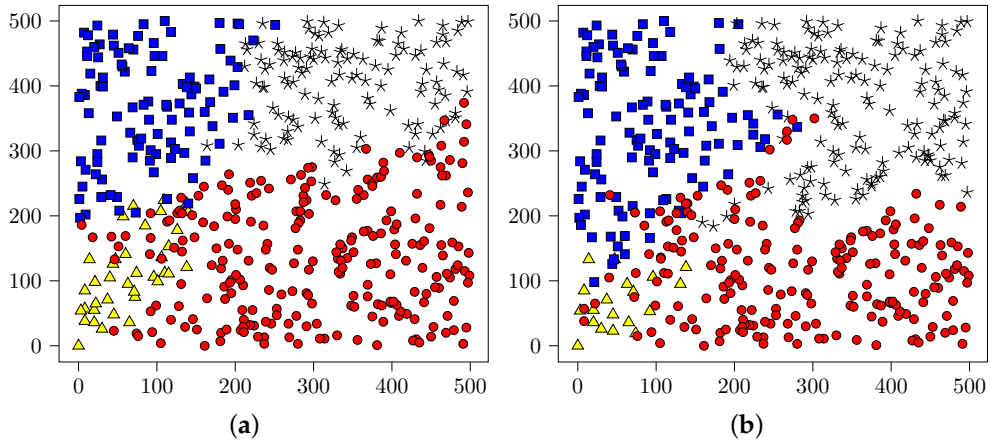


Figure 10. Node–cluster distribution when $T = 100$, $N = 500$, $L = 500$. Cluster 1: yellow triangles, Cluster 2: blue squares, Cluster 3: black stars, Cluster 4: red circles. (**a**) ICS scheme; (**b**) ACS scheme.

Impact of the UAV speed. Our scheme only plans for the UAV to visit the CHs instead of every device. When setting up communication with a device, in reality, it may require the UAV to slow down or hover still for reliable communication. However, as mentioned above, we can think of v as the average speed, considering both the hovering time and acceleration/deceleration times. In addition, the number of CHs is small; therefore, we exclude this time and assume the UAV flies continuously.

Nevertheless, we believe that it is worth observing the impact of the speed on the final aggregation delay. Figure 11 displays an aggregation delay trend when v varies in the range $[0.5, 2.5]$ m/time slot. These speeds are equivalent to $[18, 90]$ km/h. The higher the UAV speed, the more ACS outperforms other schemes. We do not simulate the UAV speed beyond that range because 90 km/h is already high.

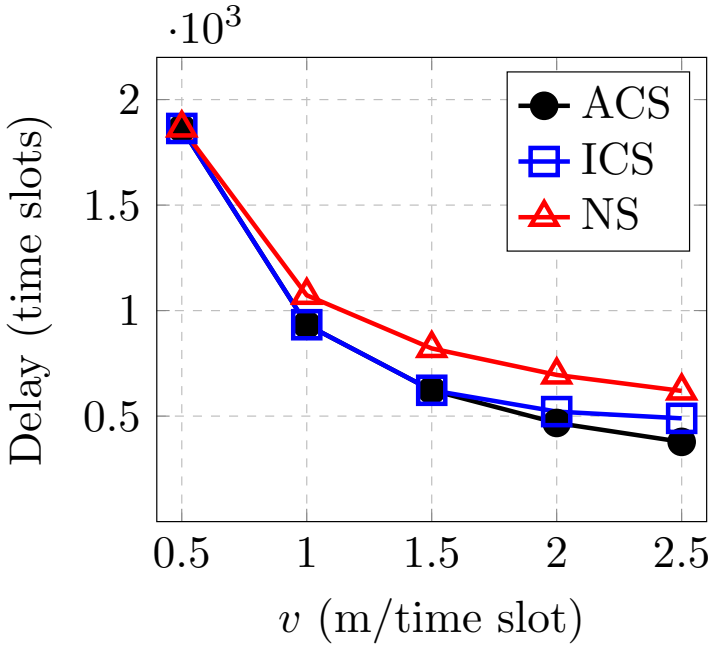


Figure 11. Impact of the UAV speed. $T = 50$, $N = 700$, $L = 500$.

Docking station departure time α_0. Recall that α_0 is the time the UAV departs from the docking station. Because the UAV flies continuously, its itinerary finishes within a constant time. As a result, α_0 determines the aggregation delay. In addition, α_0 reflects the difference between the aggregation time in Cluster 1, and the flying time of the UAV from the docking station to CH c_1. Figure 12 shows the docking-departure time of the UAV. A larger T results in a higher aggregation delay in each cluster; therefore, the waiting time at the docking station is longer. In all the cases, ACS lets the UAV take off earlier than the other two schemes. If v decreases, then the UAV needs more time to travel to the CH_1. Consequently, it needs to take off earlier. For example, when $T = 100$ slots and the scheduling algorithm is ACS, the UAV needs to start at time slot $\alpha_0 = 0$ if it flies with the $v = 1$ m/time slot (Figure 12a), while with the $v = 2$ m/time slot, it can wait until time slot 100 (Figure 12b).

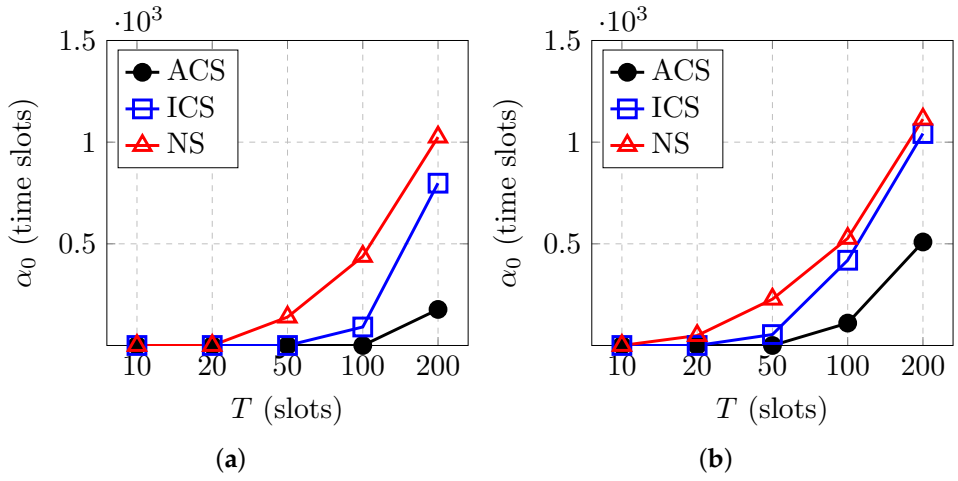


Figure 12. Docking station departure time (α_0) with $N = 700$, $L = 500$. (**a**) $v = 1$ m/time slot; (**b**) $v = 2$ m/time slot.

7. Conclusions

This paper proposes ACS, an adaptive clustering and scheduling scheme that efficiently arranges the device transmissions and UAV flying schedule. In particular, along the UAV flying trajectory, later-visited CHs should have larger allocated data aggregation deadlines. Upon calculating all deadlines at the CHs, ACS performs a reverse scheduling algorithm to iteratively grow multiple aggregation trees rooted at the CHs. The transmission schedule and sender–receiver selections are pipelined according to the device's active time slots.

Simulation results confirm the effectiveness of ACS, as it reduces aggregation time by up to 35% compared to other benchmarking schemes. ACS shows a significant improvement over ICS when the working period length is average, i.e., in this case, $T = 50$. The performances of ACS and ICS tend to converge when T is too small or too large. In the future, we intend to optimize the number of CHs and their placement to achieve shorter aggregation times. We also plan to evaluate the impact of various factors, such as fading and the signal-to-interference and noise ratio, on performance. Finally, more scenarios will be investigated, including disconnected regions, larger and denser networks, and deadline-constrained aggregation tasks.

Author Contributions: Study design, T.-D.N.; Conceptualization, T.P.V.; Data collection, T.-D.N.; Data analysis, T.-D.N. and D.-T.L.; Methodology, D.-T.L. and H.C.; Validation, T.P.V.; Writing—original draft, T.-D.N.; Writing—review & editing, D.-T.L. and H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Korea government (MSIT), IITP, Korea, in part by the ICT Creative Consilience program (IITP-2024-2020-0-01821, 30%); in part by the Artificial Intelligence Graduate School Program, Sungkyunkwan University (2019-0-00421, 10%); in part by the Artificial Intelligence Innovation Hub (RS-2021-II212068, 10%); in part by the Development of 6G Network Integrated Intelligence Plane Technologies (RS-2024-00392332, 20%); and by the National Research Foundation of Korea (NRF) (RS-2024-00343255, 30%).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Adam, M.S.; Anisi, M.H.; Ali, I. Object tracking sensor networks in smart cities: Taxonomy, architecture, applications, research challenges and future directions. *Future Gener. Comput. Syst.* **2020**, *107*, 909–923. [CrossRef]
2. Li, W.; Wang, Y.; Sun, Y.; Mao, J. Research on Low-energy Adaptive Clustering Hierarchy Protocol based on Multi-objective Coupling Algorithm. *KSII Trans. Internet Inf. Syst. (TIIS)* **2020**, *14*, 1437–1459.
3. Nguyen, T.D.; Le, D.T.; Vo, V.V.; Kim, M.; Choo, H. Fast Sensory Data Aggregation in IoT Networks: Collision-Resistant Dynamic Approach. *IEEE Internet Things J.* **2020**, *8* , 766–777. [CrossRef]
4. Zhan, C.; Hu, H.; Sui, X.; Liu, Z.; Niyato, D. Completion time and energy optimization in the uav-enabled mobile-edge computing system. *IEEE Internet Things J.* **2020**, *7*, 7808–7822. [CrossRef]
5. Bouhamed, O.; Ghazzai, H.; Besbes, H.; Massoud, Y. A UAV-Assisted Data Collection for Wireless Sensor Networks: Autonomous Navigation and Scheduling. *IEEE Access* **2020**, *8*, 110446–110460. [CrossRef]
6. Zhan, C.; Zeng, Y.; Zhang, R. Energy-efficient data collection in UAV enabled wireless sensor network. *IEEE Wirel. Commun. Lett.* **2017**, *7*, 328–331. [CrossRef]
7. Mozaffari, M.; Saad, W.; Bennis, M.; Debbah, M. Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 7574–7589. [CrossRef]
8. Liu, J.; Wang, X.; Bai, B.; Dai, H. Age-optimal trajectory planning for UAV-assisted data collection. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, USA, 15–19 April 2018; IEEE: New York, NY, USA, 2018; pp. 553–558.
9. Samir, M.; Sharafeddine, S.; Assi, C.M.; Nguyen, T.M.; Ghrayeb, A. UAV trajectory planning for data collection from time-constrained IoT devices. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 34–46. [CrossRef]
10. Ebrahimi, D.; Sharafeddine, S.; Ho, P.H.; Assi, C. UAV-aided projection-based compressive data gathering in wireless sensor networks. *IEEE Internet Things J.* **2018**, *6*, 1893–1905. [CrossRef]

11. Al-Kaseem, B.R.; Taha, Z.K.; Abdulmajeed, S.W.; Al-Raweshidy, H.S. Optimized energy–efficient path planning strategy in WSN with multiple Mobile sinks. *IEEE Access* **2021**, *9*, 82833–82847. [CrossRef]
12. Ma, X.; Kacimi, R.; Dhaou, R. Fairness-aware UAV-assisted data collection in mobile wireless sensor networks. In Proceedings of the 2016 International Wireless Communications and Mobile Computing Conference (IWCMC), Paphos, Cyprus, 5–9 September 2016; IEEE: New York, NY, USA, 2016; pp. 995–1001.
13. Say, S.; Inata, H.; Liu, J.; Shimamoto, S. Priority-based data gathering framework in UAV-assisted wireless sensor networks. *IEEE Sens. J.* **2016**, *16*, 5785–5794. [CrossRef]
14. Li, J.; Zhao, H.; Wang, H.; Gu, F.; Wei, J.; Yin, H.; Ren, B. Joint optimization on trajectory, altitude, velocity, and link scheduling for minimum mission time in UAV-aided data collection. *IEEE Internet Things J.* **2019**, *7*, 1464–1475. [CrossRef]
15. Tong, P.; Liu, J.; Wang, X.; Bai, B.; Dai, H. Deep Reinforcement Learning for Efficient Data Collection in UAV-Aided Internet of Things. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; IEEE: New York, NY, USA, 2020; pp. 1–6.
16. Zorbas, D.; Pugliese, L.D.P.; Razafindralambo, T.; Guerriero, F. Optimal drone placement and cost-efficient target coverage. *J. Netw. Comput. Appl.* **2016**, *75*, 16–31. [CrossRef]
17. Ghorbel, M.B.; Rodríguez-Duarte, D.; Ghazzai, H.; Hossain, M.J.; Menouar, H. Joint position and travel path optimization for energy efficient wireless data gathering using unmanned aerial vehicles. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2165–2175. [CrossRef]
18. Zhang, B.; Liu, C.H.; Tang, J.; Xu, Z.; Ma, J.; Wang, W. Learning-based energy-efficient data collection by unmanned vehicles in smart cities. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1666–1676. [CrossRef]
19. Zhang, R.; Pan, J.; Xie, D.; Wang, F. NDCMC: A hybrid data collection approach for large-scale WSNs using mobile element and hierarchical clustering. *IEEE Internet Things J.* **2015**, *3*, 533–543. [CrossRef]
20. Mozaffari, M.; Saad, W.; Bennis, M.; Debbah, M. Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 3949–3963. [CrossRef]
21. Mozaffari, M.; Saad, W.; Bennis, M.; Debbah, M. Mobile Internet of Things: Can UAVs provide an energy-efficient mobile architecture? In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; IEEE: New York, NY, USA, 2016; pp. 1–6.
22. Ho, D.T.; Grøtli, E.I.; Sujit, P.; Johansen, T.A.; Sousa, J.B. Optimization of wireless sensor network and UAV data acquisition. *J. Intell. Robot. Syst.* **2015**, *78*, 159–179. [CrossRef]
23. Lee, S.; Younis, M.; Anglin, B.; Lee, M. LEEF: Latency and energy efficient federation of disjoint wireless sensor segments. *Ad Hoc Netw.* **2018**, *71*, 88–103. [CrossRef]
24. Chen, X.; Hu, X.; Zhu, J. Minimum data aggregation time problem in wireless sensor networks. In Proceedings of the International Conference on Mobile Ad-hoc and Sensor Networks, Washington, DC, USA, 7 November 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 133–142.
25. Nguyen, T.D.; Le, D.T.; Choo, H. Sensory Data Aggregation in Internet of Things: Period-driven Pipeline Scheduling Approach. *IEEE Trans. Mob. Comput.* **2021**, *21*, 3326–3341. [CrossRef]
26. Nguyen, T.D.; Le, D.T.; Pham-Van, N.; Choo, H.; Van, T.P. UAV-aided Sensory Data Aggregation: Incremental Clustering and Scheduling Approach. In Proceedings of the 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Republic of Korea, 4–6 January 2021; IEEE: New York, NY, USA, 2021; pp. 1–5.