*electronics*

MDPI

*Article*

# Voting-Based Scheme for Leader Election in Lead-Follow UAV Swarm with Constrained Communication

**Yuan Zuo, Wen Yao *, Qiang Chang, Xiaozhou Zhu, Jianjun Gui and Jiangyi Qin**

Unmanned Systems Research Center, National Innovation Institute of Defense Technology, Academy of Military Science, Beijing 100071, China; yuan.zuo@hotmail.com (Y.Z.); changqiang_usrc@outlook.com (Q.C.); zhuxiaozhou.niidt@nudt.edu.cn (X.Z.); jianjungui@nudt.edu.cn (J.G.); qjyacmilan@163.com (J.Q.)
* Correspondence: wendy0782@126.com

**Abstract:** The recent advances in unmanned aerial vehicles (UAVs) enormously improve their utility and expand their application scope. The UAV and swarm implementation further prevail in Smart City practices with the aid of edge computing and urban Internet of Things. The lead–follow formation in UAV swarm is an important organization means and has been adopted in diverse exercises, for its efficiency and ease of control. However, the reliability of centralization makes the entire swarm system in risk of collapse and instability, if a fatal fault incident happens in the leader. The motivation is to build a mechanism helping the distributed swarm recover from possible failures. Existing ways include assigning definite backups, temporary clustering and traversing to select a new leader are traditional ways that lack flexibility and adaptability. In this article, we propose a voting-based leader election scheme inspired by the Raft method in distributed computation consensus to solve the problem. We further discuss the impact of communication conditions imposed on the decentralized voting process by implementing a network resource pool. To dynamically evaluate UAV individuals, we outline measurement design principles and provide a realizable calculation example. Lastly but not least, empirical simulation results manifest better performance than the Raft-based method. Our voting-based approach exhibits advantages and is a promising way for quick regrouping and fault recovery in lead–follow swarms.

**Keywords:** smart city; UAV swarm; lead–follow; leader election; distributed consensus



**Citation:** Zuo, Y.; Yao, W.; Chang, Q.; Zhu, X.; Gui, J.; Qin, J. Voting-Based Scheme for Leader Election in Lead-Follow UAV Swarm with Constrained Communication. *Electronics* **2022**, *11*, 2143. https://doi.org/10.3390/electronics11142143

Academic Editors: Zhiwei Zhao, Jorge Ortiz and Guohao Lan

Received: 15 June 2022
Accepted: 6 July 2022
Published: 8 July 2022



## 1. Introduction

The trend of global urbanization is seemingly increasing as the world population is expected to double in the next few decades [1,2]. It is undoubted that citizens and governors are now concentrating on multiple fields of cities, i.e., indispensable management [3], sustainable development [4], trustable security [2], reliable environment protection [5], etc. The smart city concept [6,7] is raised for the above issues and has been evolving [8] for decades. The recent development of Internet of Things (IoT) [9,10], fifth-generation (5G) communication [11] and multi-access edge computing (MEC) techniques [12,13] further stimulate implementations and applications for efficient city operation, for instance information framework [4], anomaly identification [14] and cyber-threat detection [15].

Although advancing technologies empower modern smart city schemes, drawbacks and challenges exist in several components of smart city operations [1,3]. For example, the transportation system in cities [16] is one of special interest. To relieve traffic congestions in various periods, massive amounts of sensors, cameras and other equipments should be widely deployed to gather abundant data and gain a global view for subsequent deep analysis [3]. This inevitably induces massive cost and numerous static deployments fail to collect real-time data [1]. Another case would be efficient communication and networking. One may argue that the 5G technique enriches mobile applications and accelerates the emergence of diverse scenarios, promoting the potential of current and future solutions [17,18]. Despite some efforts having been made to improve the availability and quality of experience (QoE) [19,20], the placement and cost issues would still affect adaptation to

dramatically dynamic implementations [11,21], i.e., natural catastrophes [22] and military events [23].

Now the unmanned aerial vehicles (UAVs) or drones-enabled scheme is drawing increasing attention and is generally accepted to be a promising way to alleviate some difficulties, such as relayed long-range communication [1,24], air-quality sensing [5], traffic improving [3,16] and information sharing in battlegrounds [23]. In effect, unmanned aerial vehicles (UAVs) have high potential, nevertheless one single UAV may have limitations in energy, sensor loads, computing and communication capacities [25,26]. Multiple UAVs, which can be called UAV swarm in a large-scale number, are put forward to cooperatively accomplish tasks that a single drone fails to tackle [27,28]. One high-performance way to organize UAVs is the lead–follow formation. It appoints a UAV to acting as leader, who takes charge of spreading unambiguous instructions and coordinating behaviors of subordinates [22,29,30], known as a *lead–follow* manner. The rest of swarm members are thus called followers. The focus of this article is on the lead–follow UAV swarm manner.

The limitation of lead–follow is straightforward. As a leader is centralized node, it is also a fragile node, which make the whole swarm fragile [23,31]. Notice that we use "*fragile*" with a global perspective to describe one that can impose influence on all other entities. In other words, if a leader cannot continue its functionality because of crashing, malfunctioning or being faulty, the swarm probably falls apart and causes a holistic swarm fault. In the lead–follow manner, a failure of the leader is likely to cause collapse and chaos of the swarm system. The lead-election is to guarantee the fault tolerance and adapt to complex, unexpected and even potentially hostile circumstances. An intuitive and viable approach to guarantee swarm working is to rapidly determine a back-up UAV to tolerate exceptions. It is nonetheless static and cannot find the optimal one based on real situations. The conventional way could be presetting a series of successors, which seems to be helpful in changeless environment. However, there might be some problems. The predetermined method is probably not able to foresee the highly uncertain situations and may not result in a suitable leader for a given task. For instance, one individual is more likely to be elected if it happens to be within the task region, or occupy the full information of targets, especially in complex and harsh battlegrounds [23]. Another case could be that a new leader should be close enough to the ground stations for forwarding of appropriate instructions, communication relay and orchestration [22,24]. Therefore, there needs to be an elastic and adjustable way to encourage the rest to make a decision [31]. The goal of this research is to make the rest of the collective UAVs recover from the systematic faults. We manage to construct an effective voting-based algorithm to discover an optimal drone to be the leader in condition that the original leader is disabled and all UAVs are decentralized. The entire process is called leader election.

It is also notable that the voting mechanism will benefit the applications in an aquatic environment. Nowadays, two major types of vehicles have drawn immense attention in unmanned research community, Unmanned Underwater Vehicles (UUV) and Unmanned Surface Vehicles (USV), respectively [32]. Various unmanned vehicles occasionally can be jointly called Unmanned Aerial and Aquatic Vehicles (UAAV) [33]. Since the UAAV network shares vital common features, some efforts are made to investigate the feasibility of method transferring [33]. In general, the acoustic communication is employed in the UUV, such as a sonar system, due to the absorption of radio signal under water [34]. For the USV, two signal transportation media are available, acoustic modem underwater and wireless access through air [35]. Apparently, the underlying network performance could be strictly limited, i.e., bandwidth, data rate, throughput, range, etc. To enable swarming of UUV and USV, distributed approaches are essential and indispensable for coordination and cooperation, where the voting-based method will play an active role. At the same time, the UAV may suffer a similar communication burden as in urban or rural situations [36]. The communication capability is always affected by the energy consumption and swarm dynamic mobility. In addition, the excessively overlapped wireless coverage may cause potential resource usage conflicts, for instance frequency spectrum and bandwidth. The proposed voting algorithm can thus be useful and applicable in a UAV lead–follow swarm.

As for the election process, there are several important concerns. Firstly, since the original leader is down, all followers shall make the decision together to approve a leader in a fully decentralized mode. A negotiation rule should be made, such that a group of independent and autonomous drones could seek out the most suitable member acting as the leader role. Secondly, all followers are assumed to have a limited communication range and be able to reach local peers. Due to the locality of swarm, the only one who has a global connection to broadcast task instructions is the leader. This configuration is essential in practice, because the networking resource and energy of UAVs are constrained [37,38]. It is also worth noting that the global connection describes the wide-area communication mode in which an alive leader is working. A practical implementation for a UAV is to program two working configurations, the leader and the follower, respectively. Only the leader will take charge of coordination and possess the key elements to relieve the interference and resource competing as aforementioned. For followers, each one would activate its limited abilities. Lastly, there ought to be an elastic metric to calculate to what extent a UAV can fit in the leader duty. That is how a UAV can recognize its local condition and quantify it to be comparable. It should also work in a UAV local computing component in an acceptable time period. A metric needs to be intentionally planned to strike a balance between the availability and computation pressure.

In this article, we propose a voting-based approach to generate a new leader inspired by a classic idea called *Raft* [39] in consensus computation algorithms [40] to cover the above points. In particular, the proposed process is based on the election stage in Raft. Moreover, The consensus problem stems from distributed computing [41], similar to the one in human decision making [42], depicted as a multi-agent consensus model. In the following part, we will refer to "agent", meaning a practical UAVand drone entity. When a leader encounters failures, the follower UAV agents with local communication would automatically launch the voting process. Once a UAV agent become the new leader, it then converts to the leader operation according to the leader role setting. To validate our method, we present a series of experiments in order to analyze the expected performance improvements and gain insights of potential advantages.

The leader election is illustrated in Figure 1. In the normal operation stage, each follower agent receives a periodical daemon (also named heartbeat) signal to remain in lead–follow behavior. If the leader fails to keep its functionality, an election process will be triggered among the rest of the followers. After a voting program in the swarm agents, an appropriate agent should be elected based on a unified measurement. Once the daemon/heartbeat signal has been rebroadcasted from the new leader, the lead–follow swarm behavior is then reformed automatically. It is noteworthy that daemon signal and granted votes ought to be transmitted through a communication channel. The communication condition, namely communication range/radius, is considered and implemented by a communication resource pool. Note that we mainly consider the communication range as a key factor in the research to investigate the impact on performance. Details will be explained in Section 4.

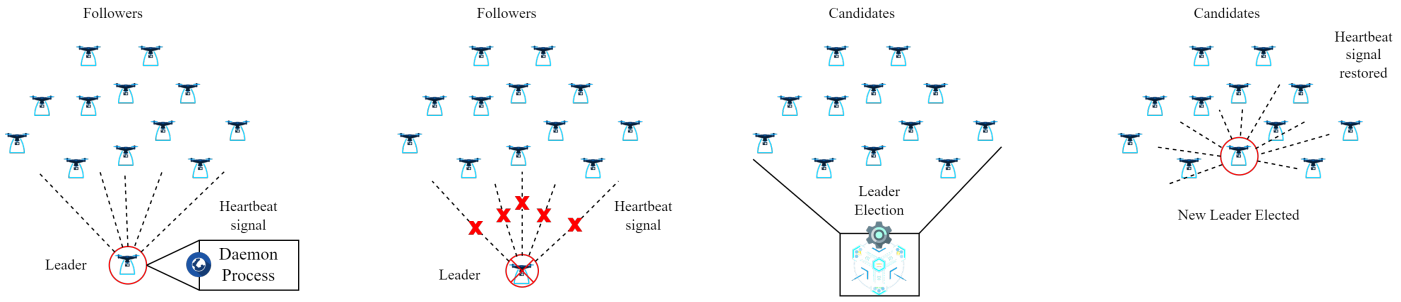



**Figure 1.** A visualization of leader-election process.

The main contributions are in the following:

- A voting-based mechanism is proposed to encourage swarm members to produce a robust and optimal leader. The proposal aims at working in a fully distributed and decentralized environment under restricted communication conditions.
- A dynamic criterion for instantly qualifying individuals is proposed, which attempts to reflect how a UAV is suitable for a simple lead–follow swarm behavior. We provide four vital design principles to make the measurement interpretable.
- To deeply analyze the impact of communication range, we devise a networking resource pool to simulate the available data exchange channels and key information sharing process.
- Experiments are conducted in four facets to broadly demonstrate empirical results. The corresponding data analysis and explanations are given to emphasize the improvements induced by the proposal.

The rest of contents are organized as belows: Section 2 outlines the related ideas and research works; Section 3 gives formulation of leader-election process; the details of the proposed approach is portrayed in Section 4, followed by the experiments in Section 5. Finally, Section 6 draws a conclusion about the work presented.

## 2. Related Works

During recent decades, unmanned aerial vehicles (UAVs) or the drones-enabled scheme have been drawing increasing attention. The common wisdom is that the drone-based scheme should be promising for solving various challenges, owing to characteristics such as flexibility, low-cost, adaptability, mobility and et al. [43]. UAVs are indeed deployed for military purposes at the beginning [23,44], while the civilians are also benefactors from their potential [45]. Drones and UAVs are helpful in smart cities with urban IoT and edges [8], such as enhancing communication [1,24], reducing air pollution [5], monitoring and managing traffic [3,16] and building robust military forces [23]. However, an single UAV unit is facing problems, for example, energy shortage, payloads restrictions, computation capacities and etc., [25,26]. Multi-UAVs and UAV swarm techniques are developed to leverage the advantages of cooperation [46].

Admittedly, swarming has advantages, and there should be interaction protocols for UAVs collaboration in swarm, owing to distribution and dispersion [47]. Efforts have been made to settle the challenges, for example decentralization [48] and hierarchical structure [49]. However, decentralization may consume plenty of communication resources and require a specific control protocol [25,50], which incurs implementation difficulty. A lead–follow formation has become a conventional and useful solution to a plenty of urban problems [22,29,30].

The consensus idea initially comes from distributed computation [40] in computer science, which principally targets fault tolerance in distributed networking environments [51], a classic problem called the "Byzantine Generals" problem. One may trace consensus in distribution back to a harmony among data consistency, system availability and partition tolerance [52]. The most recent explosion of Bitcoin leads to a research surge of consensus in blockchain [53], which is expected to verify and ensure the data integrity. There has been well-known research motivating distributed machines to reach a consensus targeting the "Byzantine Generals" problem [39,54]. Since the *Paxos* algorithm [54] is braodly believed to be difficult to understand, the *Raft* algorithm [39] is deliberately built to simplify the consensus procedure and strives to achieve a balance between performance and understandability. The Raft algorithm decompose the consensus into three subproblems: leader election, log replication and safety. Our proposal is accordingly inspired by the *leader-election* stage and modified to fit in a UAV swarm scenario.

As for the consensus in a UAV swarm, it can be viewed as all UAV agents attempting to cooperatively reach agreement on issues in terms of task decision [41]. Originally, the Boids/Reynolds model is proposed in [55], followed by a series of works in dynamic multi-agent flocking controling and collectives consensus analysis [41]. Similarly, the lead–follow swarm behavior is also considered with consensus [47]. A distributed swarm system thus

must effectively deal with orchestration, organization, interaction and so on [27,28]. On the contrary, a lead–follow enabled swarm is a centralized swarm system and is unified by nature, because only one center is responsible for commanding discrete elements. This intrinsic feature thus gives the leader an indispensable role in several tasks [56,57].

To find a proper agent to serve as the leader, some existing works focus on clustering collective units and determine a cluster head [31]. Mou Zhiyu et al. [56] investigate the hierarchical UAV swarm structure and employ a graph attention-based algorithm to detect clusters and the leaders. In [58], an energy-aware node clustering algorithm is proposed in order to extend the lifetime in a wireless sensor network, which is based on particle swarm optimization with combined objects to optimize the utility. The method merely considers two energy-related factors and has difficulty fitting in highly dynamic applications. Ma Ting et al. [57] introduce a modified k-means algorithm to select a super cluster head UAV agent with low latency, gaining an efficient swarm management. The authors in [31] are dedicated to optimal drone communication and put forward a bio-inspired cluster head selection algorithm. They endeavour to divide the whole swarm into small groups and enable the algorithm locally based on residual energy and distance. These methods are specifically designed to meet trajectory and groundstation matching demands regardless of fault and disaster recovery for the swarm. Frequent changing cluster heads probably brings about the instability of swarm and extra communication overheads. Additionally, the mutual communication capacity has not been explicitly examined. In this work, we unfold several useful measurement designing principles and propose Raft-inspired low-cost voting scheme to make the collectives obtain a deterministic leader. Ultimately, the experiments illustrate advantages of the proposal over the original Raft algorithm.

The Raft algorithm is proposed for the consensus achievement in distributed computing. There are some relevant works dedicated to swarm clustering and head selection. These works prefer a well-designed measurement calculation to determine which one is capable of handling the cluster-head role, for instance, distance- and residual energy-related fitness values are designed in [31]. In our work, the accomplishment of consensus principally attracts the attention, realized by the voting mechanism. A swarm system of unmanned vehicles can also be deemed to be a distributed computing system. Each vehicle individual resembles a computing node, as unmanned vehicles have limited computation capability and they are interconnected by an underlying network. On the other hand, the election process is one stage of the Raft algorithm and can be transplanted and deployed in other distributed systems. Therefore, the election process has potentials to adapt to an unmanned vehicle swarm. In the following experiment, we will present the effectiveness and availability of the voting approach.

## 3. Problem Setting

In lead–follow swarm behavior [30,59], the collective agents endeavor to regroup by determining which individual is able to lead the rest to finish certain tasks, if the former leader encounters a fatal crash or failure. As agents are independent, self-controlled and inter-connected, we define the regrouping activity as a voting-based leader-election process through interaction between each other. Thus the election process can be modelled and converted to consensus achievement in a distributed system [59].

Due to decentralization, follower agents can only acknowledge key information from the surroundings and spread the states to limited peers. Analogue to elections in the real world, we also assume that a leader in a swarm should possess some advantages in terms of particular transactions over others based on its characteristics. Transactions can be targets of tasks [30], pre-defined order and etc. Then the goal is to find the optimal agent with the highest quantified characteristics.

To further digest the routine, we shall formulize the above as an maximization issue. Firstly, let $\vec{x_i}$ be the characteristics of agent $i$ in swarm $\boldsymbol{A_s}$ representing the current status:

$$\vec{x_i}^t = (i, \overrightarrow{Pos}^t, \overrightarrow{Vel}^t, E^t_{res}, Type, Task^t), \tag{1}$$

where time information $t$ is attached. $\overrightarrow{Pos}^t$ and $\overrightarrow{Vel}^t$ denote the position and velocity of $i$-th agent at time $t$, respectively. $E^t_{res}$ is the residual energy and $Type$ corresponds to the agent type. $Task^t$ would be any key elements relevant to the task payloads.

A crucial part of election is how to quantify and measure the status of an agent. The metric could consider different factors to make sure the leader is suitable for a given mission. In general, a UAV swarm is assigned with tasks/missions that directly guide the collectives towards a goal accordingly. The metric setting is supposed to be comply with the task and the corresponding goal, which could be viewed as *task-oriented*. In this article, we would like to introduce the design paradigm of qualification measure that is helpful to identify the most appropriate follower to be elected. It is of special note that the eligibility/qualification measure is task-oriented rather than intuitive. The term *task-oriented* indicates the distinct targets of a UAV swarm to expect diverse calculating configurations. For instance, the leader is inclined to be close to the ground station for communication relaying; the leader could be one that is equipped with high-performance sensors and large batteries for environmental perception, and so on. To be more detailed, we exemplify the claim by formularizing a real-value function in the following:

$$QualifValue = \mathcal{F}(\Phi_1(\overrightarrow{Pos}), \Phi_2(\overrightarrow{Vel}), \Phi_3(E_{res}), \Phi_3(Task); \theta_1, \theta_2, \theta_3, \theta_4), \tag{2}$$

where $\mathcal{F}(\cdot) \in \Re$ is a real value mapping of factors in status, such as position, velocity, residual energy, task, etc., parameterized by $\theta_i, i = 1, 2, 3, 4$. The purposed of $\Phi_i(\cdot), i = 1, 2, 3, 4$ is to standardize and normalize categorical information as computational data. The parameters, $theta_i$, control the weights and influence on the combined qualification measure. We thus encourage readers to adapt the Equation (2) to a specific swarm assignment. For the sake of simplicity. A measurement function is defined as for the experiments:

$$Q^t_i = f(\vec{x}^t_i), \tag{3}$$

where $Q^t_i \in \mathbb{R}$ can be called eligibility or qualification measurement at time $t$ and $f(\cdot)$ is a customized mapping. Let $C_r$ be the communication range. In this article, we assume all UAV agents have the same communication ability. The neighbourhood $Nei^t_i$ of agent $i$ is then written as:

$$Nei^t_i = \{j | D_{eu}(j, i) \leq C_r, j \in \boldsymbol{A_s}\}, \tag{4}$$

where, $D_{eu}(\cdot)$ is the Euclidean distance. Note that $i \in Nei_i$ for all time. Apparantly, we have:

$$\begin{aligned} i, j \in Nei^t_i \bigcap Nei^t_j, i \neq j \\ \text{if } D_{eu}(i, j) \leq C_r. \end{aligned} \tag{5}$$

Similarly, if agent $i$ cannot connect agent $j$,

$$\begin{aligned} i, j \notin Nei^t_i \bigcap Nei^t_j, \\ \text{if } D_{eu}(i, j) > C_r. \end{aligned} \tag{6}$$

For locality, there exists an agent with highest qualification $Q^t_i$ in the neighours of agent $i$. That is

$$LocalMaxAgent^t_i = \arg\max_j(Q^t_i(Nei^t_i)). \tag{7}$$

Accordingly, our goal is to find:

$$GlobalMaxAgent^t_{|t=T} = \arg\max_i(LocalMaxAgent^t_i)_{|t=T}, i \in \boldsymbol{A_s}, \tag{8}$$

where $t = T$ implies that we concentrate on the leader election at one moment. For simplicity, we can omit the time symbol, namely:

$$GlobalMaxAgent = \arg\max_i(LocalMaxAgent_i), i \in \boldsymbol{A_s}, \tag{9}$$

The process of finding the agent with the global maximal qualification is the routine of agreement in the UAV swarm. As a swarm can be considered as a distributed computing system, we then turn the maximization of qualification into consensus on a leader by voting. The following analysis clearly elucidates that the voting-based election approach is able to assist the independent UAV agents in a swarm system to reach agreement and collaboratively generate an acceptable leader.

## 4. Proposed Method

In this section, the detail of our proposed algorithm is illustrated. As aforementioned, the proposed algorithm is inspired by the leader-election stage of the Raft algorithm [39] that has been implemented as the foundation of distbuted coordination mechanism.

### *4.1. The Leader Election Process*

The leader election scheme in UAV swarm generally employs a similar process to that of the Raft algorithm in order to reach an agreement. At the beginning, a leader UAV (called a leader agent in simulation) can be appointed in a swarm based on the incoming tasks, preference of an operator, specific demands or randomly. After the startup, it is assumed that the UAV swarm turns to conduct predefined tasks in a lead–follow manner.

From the view of a leader, it is not only responsible for working coordination through communication, but also keeps daemon module alive and periodically broadcasts heartbeat signals to manage the whole swarm system. Whenever there exists an event making the swarm lose its leader, i.e., crashing or fatal malfunctioning, the leader-election process would take place subsequently.

From the perspective of followers, one may expect to receive the heartbeat signal to maintain the leader-follower relationship. A short-term waiting process with a countdown clock is setup in each follower to determine whether they are losing contact with the leader. When a heartbeat message is detected, the clock is reset for a new waiting time slot. Once the heartbeat signal is terminated and the waiting process finally ends with the clock *timeout*, a leader-election process will be instantly triggered to reform the lead–follow swarming manner. The scheme described above can be regarded as a state-transition process, commonly including the follower state, the candidate state and the terminal leader state if elected.

A comprehensive process diagram can be seen in Figure 2. UAV agents start from the *Normal Running Loop* and emerge as lead–follow swarming. After the leader crashes, the election process in each agent is then activated. Followers convert themseleves into candidates which conduct a voting operation. In the voting operation, each individual calculates a *qualification measure* for itself and accordingly votes for the highest one becoming the new leader. Lastly, the lead–follow swarming has been reformed.

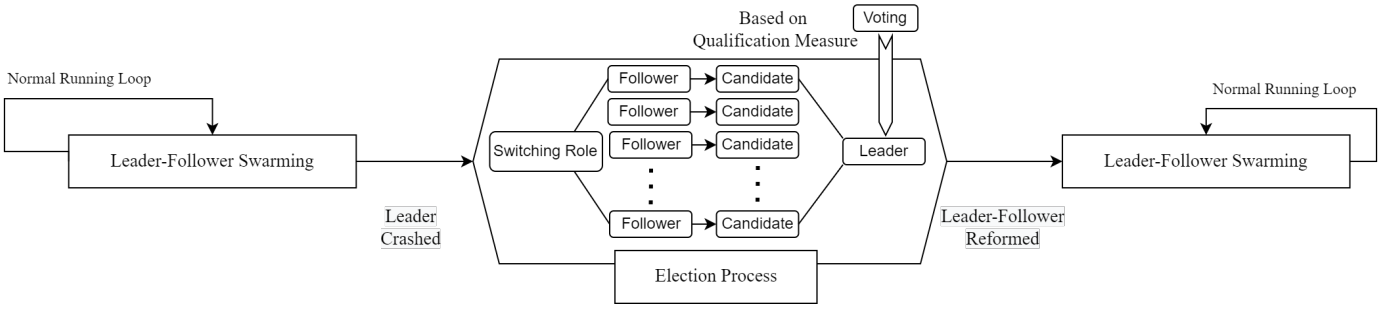



**Figure 2.** A comprehensive leader election process diagram.

### *4.2. Qualification Measurement*

The qualification measurement is an important element for the proposed leader election method. It is clear that in the Raft algorithm, a leader of a computing cluster is the one with the latest (up-to-date) log index. A vote can be granted if and only if the candidate log in a request message is ahead of the receiver. The fixed mechanism and criterion makes the results deterministic. Although there is chance that no majority is produced for spilt votes, a re-election method is employed with randomly-timed requesting.

However, the fixed criterion is a method that a UAV swarm attempts to avoid. Since independent and equal agents expect to find the most appropriate individual for the leader position in a task, a deterministic appointment would merely generate sub-optimal one. As such, there should be one *value* that is able to evaluate the current status of each individual. The *value* indicates to what extent an agent is suitable against one certain task or target for the leader role.

The *value* mentioned above is defined as qualification measure or eligibility value. The qualification measure should be able to dynamically and accurately capture the inherent property of an object. The qualification measure must be designed accordingly and specifically to the object of interest.

In this article, we mainly focus on the design paradigm and the proposed voting method. The qualification equations are simplified in order to make the simulated agents comparable. For the sake of simplicity, we devise the qualification mainly based on a distance from a virtual center position:

$$\mathcal{F}(\cdot) = \theta_1 \cdot \Phi_1(\overrightarrow{Pos}), \tag{10}$$

where, $\theta_1 = 1, \theta_2, \theta_3, \theta_4 = 0$ for Equation (2),

$$\Phi_1(\overrightarrow{Pos}) = \frac{1}{||P_i - P_{vm}||_2}, \tag{11}$$

$$\mathcal{F}(\cdot) = Q_i = \frac{1}{||P_i - P_{vm}||_2}, \tag{12}$$

in which $Q_i$ is the qualification measure value of $i$-th UAV agent. $||\cdot||_2$ refers to Euclidean distance, and $P_i$ and $P_{vm}$ indicate the position of $i$-th UAV agent and the virtual center of all agents, respectively. A simplified example is illustrated in Figure 3. The red agent is the closest one and would been elected as a new leader. Since the agent is active in the Euclidean space and $||\cdot||_2$ is utilized to calculate the distance, the position vector $\overrightarrow{Pos}$ can comprise of latitude, longitude, and altitude. For the sake of simplicity, we manually fix all running agents the same altitude in the experiments, where three-dimensional position information is also helpful and applicable.

Provided that each agent has a limited communication range (denoted by communication radius), an assumption is that the center can connect as many neighbours as possible. The distance-based qualification set as above tend to make the one closest to the center to be the leader. Here, the virtual center information could be delivered by the previous leader before failures occur. In this article, we simply make the virtual center information to be transmitted from the previous leader along with heartbeat signal to exemplify the whole process. Other plausible implementations are also encouraged and should comply with some certain tasks. The intuitive way to calculate the eligibility of agents can vary with dynamical circumstances and mark the most qualified (only in terms of our setting) agent instantly.

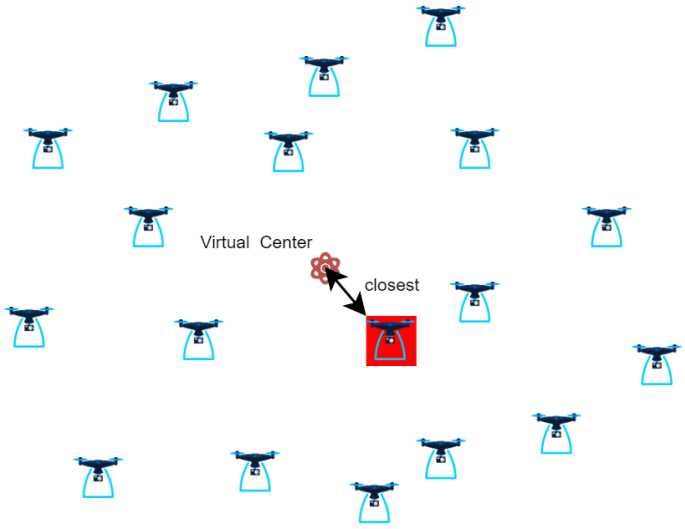



**Figure 3.** An example of the utilized qualification calculation.

It is of particular note that there might not exist a universal measurement standard, and one devised setting of a swarm must adapt itself to specific situations, tasks and goals. Here, we summarize four seemingly useful principles for assisting criterion design. The qualification could:

- Not remain constant;
- Evaluate how close each agent is to the target of interest;
- Distinguish agents only based their local status and properties;
- Be as simple as possible to calculate instantly or in as short a time as a swarm can tolerate.

It is obvious that the four principles are intuitive and not compulsory. We hereby encourage readers to adjust the design procedure according to their own scenarios for availabilty and reliability. A potential and feasible way is to turn qualification into an optimization problem instead of elaborately and manually planned if the target is vague and uncertain.

### *4.3. The Proposed Election Protocol*

The proposed election protocol employs the state-transition similar to *Raft* as agents role-transition mode in a UAV swarm. An illustration can be viewed in Figure 4.

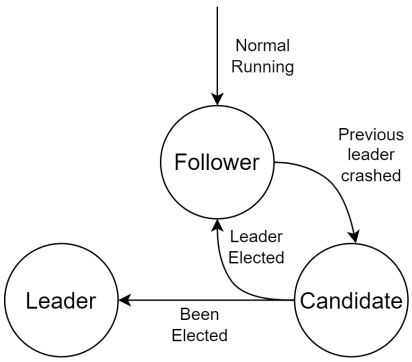



**Figure 4.** The role transition of swarm agents.

The role-transition explicitly shows clear and effective operation logic, which guarantees stability and availability of the UAV swarm. Differing from the *Raft* algorithm, there is no term limit for the leader. In the *Raft* algorithm, a term servers as *logic clock* to discover whether logs in leader is obsolete, and the leader position shall be given back to

the up-to-date computing node. The discovery task will inevitably consume networking resources and bring about extra communication overhead. In a UAV swarm, the leader role is inclined to be permanent, as the swarm operation hierarchy should stay fixed and stable during task execution, and maintain a low level of networking traffic load.

Figure 5 shows a complete program flow diagram. Each individual agent will execute following the program flow independently. In initialization, all agents enter into the follower role except one pre-defined leader. The periodic heartbeat signal of the leader proclaims its authority and other agents will keep their role. Once the heartbeat is lost, agents will wait for reconnection until *timeout*. After changing to a candidate role, agents instantly synthesize their own qualification value based on pre-defined equations, following the method described in Section 4.2.

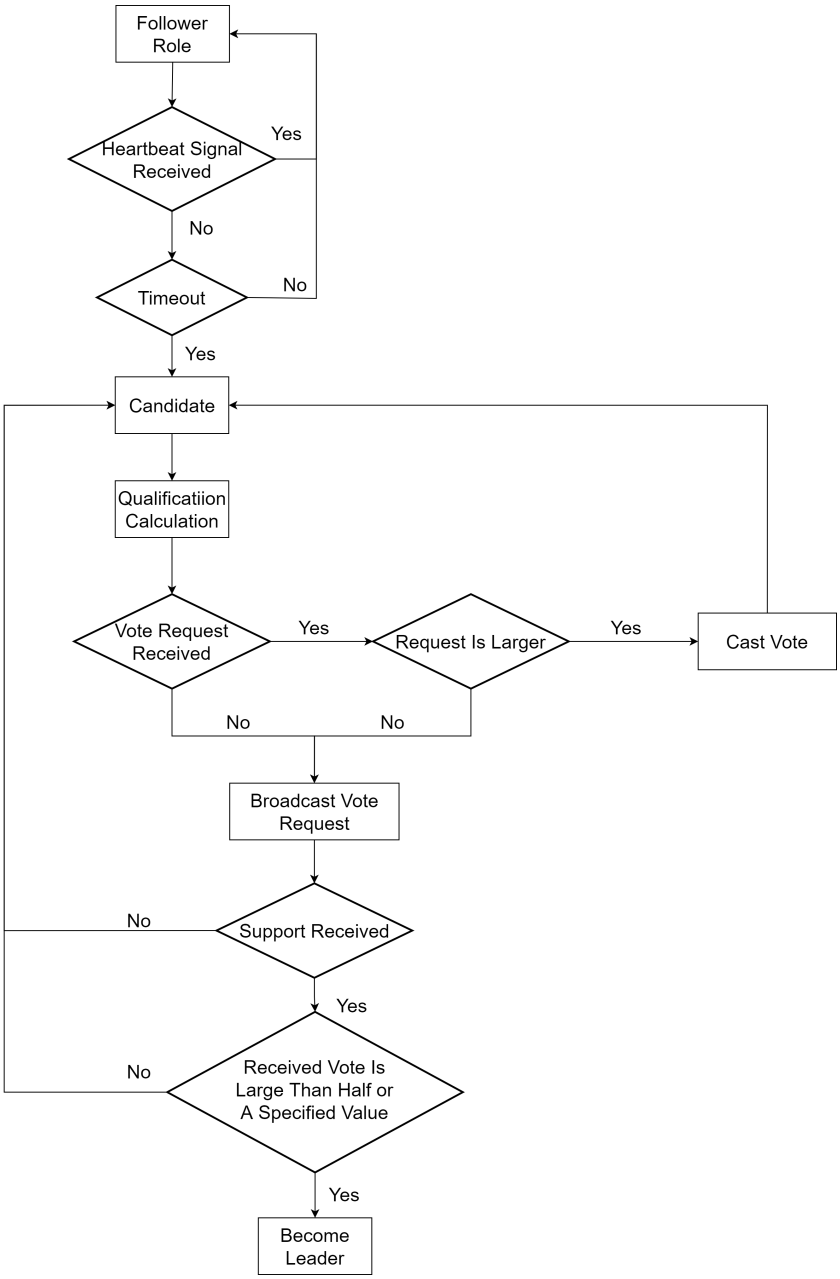



**Figure 5.** The program flow diagram.

The next step is to check whether vote requests have been received and all requests are collected in a list. One may compare their own qualification with those incoming information from others. Being aware if there exists a higher qualification value, agents

automatically cast votes to the individual with the highest value and restart the candidate operation. If the above two statements are false, agents shall broadcast vote requests to neighbours inside their communication range (denoted by communication radius). A short-time duration is set to aggregate all replied votes, followed by the step where the support votes count is compared with a threshold. The threshold can be pre-loaded as the half of total number or a specified value. A leader is thus identified once the aggregated votes exceed the threshold. Otherwise, one agent would return the beginning of candidate operation loop and calculate the qualification value to keep it up-to-date.

Here, the consensus-based leader election algorithm is separated as three parts: Leader operation, Candidate operation, and Follower operation. Algorithm 1 presents the global loop running in agents and the role has been initialized as leader or follower for each individual. The *Communication Module* processes essential exchanged messages, i.e., heartbeat signal, requests, votes and etc., followed by three main parts.

**Algorithm 1:** Global loop

```
Initialization (Leader, Follower);
while True do
    Communication Module;
    if Leader role then
        Leader loop;
    if Follower role then
        Follower loop;
    if Candidate role then
        Candidate loop;
```

The leader and follower loop in Algorithms 2 and 3 is clear. The leader could retain its authority or actively quit due to fatal failure. The follower either continuously handles instructions from a leader or the timeout process occurs and it converts to the candidate role.

**Algorithm 2:** Leader Loop

```
Input: Tasks to be executed
Output: UAV actions
Any specified UAV agent can be initialized as leader role;
if fatal failure existence then
    Quit leader role;
    Dispatch from swarm;
    Break;
if quit leader order from higher priority then
    Quit leader role;
    Break;
Broadcast heartbeat signal;
Task execution process;
```

In the candidate loop, see Algorithm 4, it would firstly check if another agent claims winning the election. If not, the algorithm then keeps track of the running flow along with the diagram in Figure 5. Note that one significant difference between our method and *Raft* is that distributed nodes cast only once during one given term. Consequently, there is the possibility that votes are split and a re-election is then enabled. A randomized latent is employed for a next-round vote request and the corresponding term is increased. The re-election with a random latency is of importance for the Raft algorithm to avoid an impasse of no majority winning. The re-election could generate a similar leader if and only if the most suitable candidate claims its authority in an early stage. On the other hand, the re-election might produce a distinct leader because the environment is uncertain, making

the qualification value inconstant. The proposal in this work has cancelled the re-election procedure in contrast to the Raft. In our work, we consider that the environment is ever-evolving and dynamic and the swarm can quickly react to those unexpected changing. That is, one member inside the swarm should adaptively decide to vote for a new candidate immediately. In our method, only one vote is permitted at a time but there can be more than one casting during election, since the flexible qualification value could always reflect the most proper individual. We regard this as being of a *regret* fashion beneficial from the dynamic qualification measurement. It implicitly allows agents could persist in their choice by multiple response, and in the other way that one could "change its mind" if a better candidate shows up. The empirical comparison can be seen in Section 5.

**Algorithm 3:** Follower Loop

```
Input: Heatbeat signal, tasks assigned in swarm
Output: UAV action
Initialized as a follower;
if no heatbeat signal then
    Timeout process;
    if timeout then
        Enter candidate loop;
else
    Follow leader instruction;
```

**Algorithm 4:** Candidate Loop

```
Input: Vote related information
Output: Election Claim
if Leader been elected then
    Break candidate loop and become a follower;
else
    Calculating qualification measure;
    if vote requests received then
        Compare qualification;
        if self qualification is less than requests then
            Cast vote;
        else
            Broadcast vote request;
            if Received votes exceed a threshold then
                Broadcast elected claim;
                End candidate loop and turn to leader role;
            else
                Continue candidiate loop;
    else
        Broadcast vote request;
        if Received votes exceed a threshold then
            Broadcast elected claim;
            End candidate loop and turn to leader role;
        else
            Continue candidiate loop;
```

Since the original paper of *Raft* [39] has not presented the complexity analysis, we hence endeavour to compare the *Raft* algorithm with our proposal. The *Raft* method consists of several phrases to complete distributed consensus, while only the leader-election

phrase is situated in our scope of interest. As a matter of fact, the FLP impossibility [51] elucidates the difficulty in obtaining consensus with limited termination time under some circumstances. Thus, we mainly focus on the voting computation instead of the ultimate consensus achievement. According to [39], one computing node either votes for the largest term count or accumulate incoming supports. Suppose there are $\frac{1}{C_v^R} \cdot n$ incoming votes, the computation complexity would be $O(\frac{1}{C_v^R} \cdot n) \iff O(n)$. In convention, the complexity of the additive operation is $O(1)$. $C_v^R$ is a non-zero constant and $n$ is the number of all individuals. If $C_v^R = 1$, one could receive all votes including itself. If $C_v^R > 1$, one may have partial votes. Note that the proposed approach in this article enables one individual to alter its choice following the basic voting accumulation. Let $C_{cm}^{ppl}$ and $C_v^{ppl} \cdot n$ be the counts of "changing mind" and incoming votes, respectively. The complexity then becomes $O(C_{cm}^{ppl} \cdot \frac{1}{C_v^{ppl}} \cdot n) \iff O(n)$, where $C_v^{ppl}$ has the similar meaning to $C_v^R$.

### *4.4. Communication Issue*

One vital issue in swarm leader election is the communication network. In common and traditional distributed computing systems, spatial distance imposes a partial limit on the inter-connection of node members. Nevertheless, spatial communication range is one of the most crucial factors of wireless swarm network, which makes the operation environment in a UAV swarm distinctive from the traditional one. In this article, we chiefly consider the communication range influence on a high application level, denoted by communication radius. To simplify the issue, an assumption is that any agents within the communication range of another would be able to exchange key information, also called neighbours.

Figure 6 explains how agents hold the connectivity. In the physical level, each agent denoted as local center has its own connectable area, called a neighborhood. In this area, all other agents are called neighbors. The whole swarm can be logically mapped into network level as functional nodes, organized by node-to-node relations, namely data links. A communication module ought to abstract the relationship inside a swarm and handle necessary information transferring by data links. Detailed settings are presented in Section 5. Note that in this article we mainly focus on the impact of communication range (radius) imposed on performance of leader election.

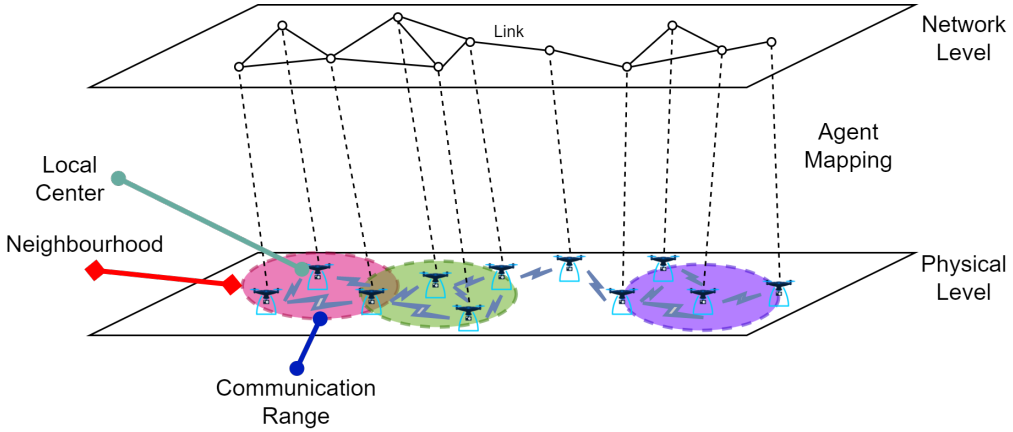


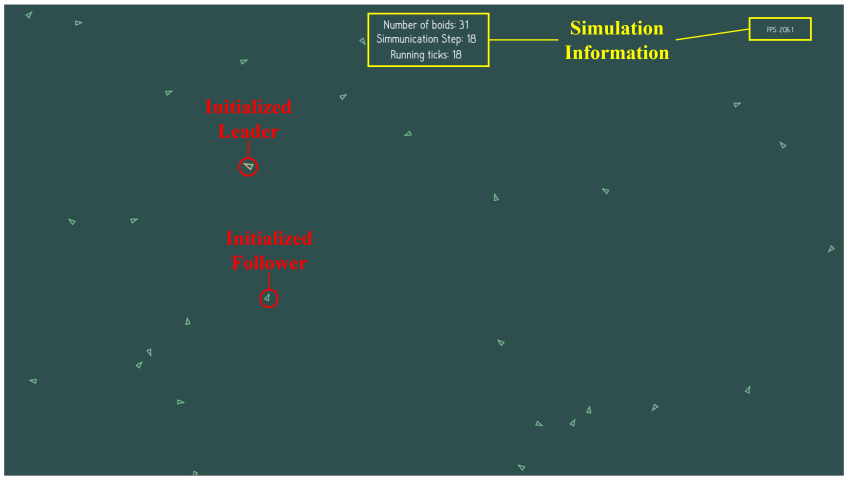

**Figure 6.** Comminucation mapping from physical world to networking level.

## 5. Experiments

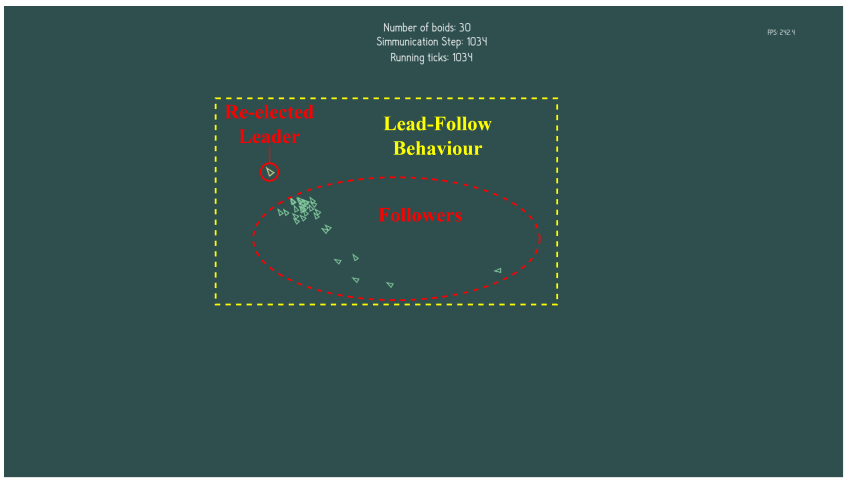
In this section, we will demonstrate empirical resutls of simulation experiments. Experiment settings are firtly presented, followed by communication module, and lastly comprehensive results are showcased.

### *5.1. Basic Settings*

Our simulation experiments are conducted and implemented based on a lightweight agent-based environment, which exemplifies a boid model and open-sourced in Github (https://github.com/florimondmanca/pyboids, 7 July 2022), called *pyboids*. The agent-based environment is built on *Pygame* framework (https://www.pygame.org, 7 July 2022), which provides a toolkit for developing games with Python. Figure 7 overviews some fundamental elements of this siumlation environment.




(**a**)




(**b**)

**Figure 7.** An overview of simulation environment setup. (**a**) Initialization of experiments. (**b**) Lead-Follow behavior of swarm.

The original simulation framework aims at implementation of classic Boid-flocking model, including random wandering, alignment, separating, coherency, border, threat evasion and lead–follow behaviors. We tailor and modify the environment to adapt to leader election scenarios with basic follower and border behaviors. As for the graphic interface, some simulation information are displayed at the center and right corner of the screen. In Figure 7a, 30 normal follower agents and 1 leader agent (total 31 agents) are initialized, colored with green and yellow, respectively. All the agents are evenly distributed in experiment space, at the beginning of a simulation round. In Figure 7b, a lead–follow running example is highlighted, where the leader is elected after the initialized one is crashed (set as disabled in simulation). Noting that in Figure 7b, it illustrates a possible swarming manner snapshot that a lead–follow formation could be after the new leader has

already been elected. The lead–follow manner is mainly controlled by the swarm policy instead of a task-oriented election measure.

Some experiment settings follows the description in Table 1. For each experiment sample, 100 simulation rounds shall be conducted to validate the results. In one simulation case, a given number (from 30 units to 100 units, increased by 10) of agents are randomly initialized and uniformly placed across the main simulation area. The communication is varied from 200 units to 700 units by stepping 25, to investigate the influence on performance. Here, one unit is an abstract term corresponding to a length unit in real world. *Leader Crashed Time* in Table 1 indicates that the initialized leader is intentionally disabled at the 50th simulation step. *Heartbeat Waiting Time* allows agents to wait for reconnecting the original leader within 10 steps. The *Max. Election Elapsed Time* term means there are at most 50 simulation steps for the swarm electing a new leader. If the election cannot be finished within the maximum election elapsed time, a failure trial is recorded.

**Table 1.** Experiment Settings.

| Attributes | Values |
|---|---|
| Number of Leader Agents | 1 |
| Number of Follower Agents | from 30 to 100, increased by 10 |
| Communication Range | from 200 to 700, increased by 25 |
| Initialization mode | uniformly spatially distributed |
| Leader Crashed Time | 50 |
| Heartbeat Waiting Time | 10 |
| Max. Election Elapsed Time | 50 |
| Simulation Round | 100 |

### 5.2. Communication Simulation

We construct a specific communication module in order to ensure the simulated inter-communication process can clearly handle data transferring issue. The communication consists of two main components, one for basic link attributes and message records, the other for all message indexing, aggregation, coordination and synchronization in a networking pool.

For a link component, a UAV agent node can merely reach neighbours falling in a closure bounded by its communication radius. Each time a node endeavours to keep contact with another node, a virtual link is thus built with essential attributes, for instance source/destination identity pair and message to be sent. The identity pair must include a unique and unchangeable token, i.e., a virtual IP address, virtual port number, virtual protocol, etc. Messages are recorded in a list-like buffer to store multiple pieces of messages.

As for the networking pool, agents may decide to distribute information to any other individuals, whereas the networking pool is responsible for determining whether a requested link can be established. At the beginning of each running loop, agents submit the link establishment requests based on their local requirements. The communication module tends to iterate over the requests collection, use low-level simulation data to detect link existence and filter out unestablished ones.

We describe the communication module in the following. We attempt to depict the implementation of the networking pool, even though the fine-grained network simulation is out of the scope of this article. Based on the two major components, each running agent will be assigned with a fixed virtual IP, denoted as unique numbers. We assume all agents share the same virtual protocol, denoted as a string. When a transmission to a given node is requested, an agent would launch a link creation with a tuple including mutual IPs and data. The new link content will be pushed into a public network stack. Secondly, the networking component takes over the subsequent process. It is responsible for emitting non-existing links and collect the message data in terms of destination. By emitting non-existing links, the networking component will maintain a neighbourhood table for each agent, recording the nodes that are within the communication range. Destination IPs will

be checked in the corresponding table. If matched, the link will be added into a temporal public link list. In the following, a hash-like table, e.g., a dictionary in Python language, will be built based on the public link list. The keys are destinations and the corresponding values contain the source IP and the message data. When an agent is activated in the next simulation step, it will acquire messages based on its IP identity from key-value pairs for the behavior update.

In each simulation update step, the communication pool is fully checked to ensure agent information exchange and emptied just before next update step to refresh the simulation inner state.

### 5.3. Empirical Results

We run the simulation to focus on the influence of communication range imposed on leader-election process in terms of four typical facets of performance: completion rate, success rate, average rank and simulation time. The *completion rate* refers to how many times out of the total simulation rounds a leader can be elected regardless of concrete ranking. The *success rate* represents how many times out of total simulations a leader can be elected with a ranking higher than a preset value, i.e., top 10%. Obviously, the *completion rate* checks the availability, whereas the *success rate* reflects the effectiveness and robustness. The *average rank* metric unfolds the quality of algorithm outcomes, and the *simulation time* gives insight to the efficiency of potential solutions.

We split our experiment into two scenarios based on the vote threshold settings, a fixed value (15 votes) and a percentage (50%, half of swarm size). That, is one agent should receive sufficient amount of votes surpassing the threhold to obtain leadership. The purpose of setting two thresholds is to verify the adaptability and robustness of the election mechanism under diverse circumstances.

#### 5.3.1. Completion Rate

An election process is considered to be finished if and only if an agent claims leadership before the *Max. Election Elapsed Time*, described in Table 1. Therefore, the completion rate is computed as:

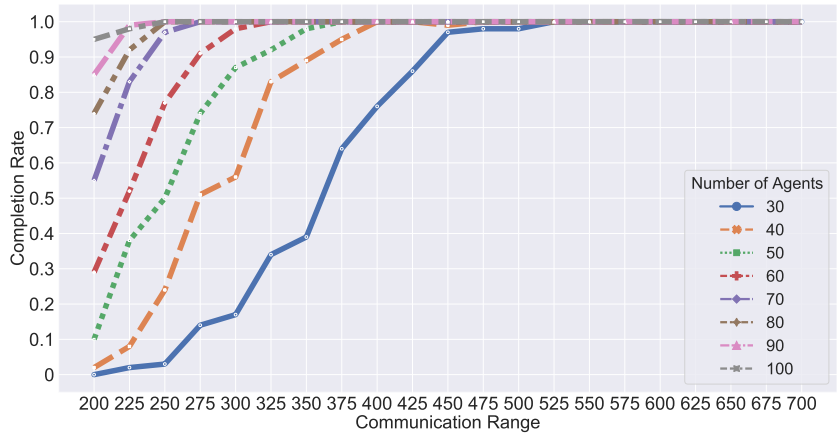
$$CompRate = \frac{C_{elected}}{C_{sim}}, \tag{13}$$

where, $C_{elected}$ is how many times a leader in swarm has been elected, and $C_{sim}$ is the count of all simulations, namely 100 as in Table 1.

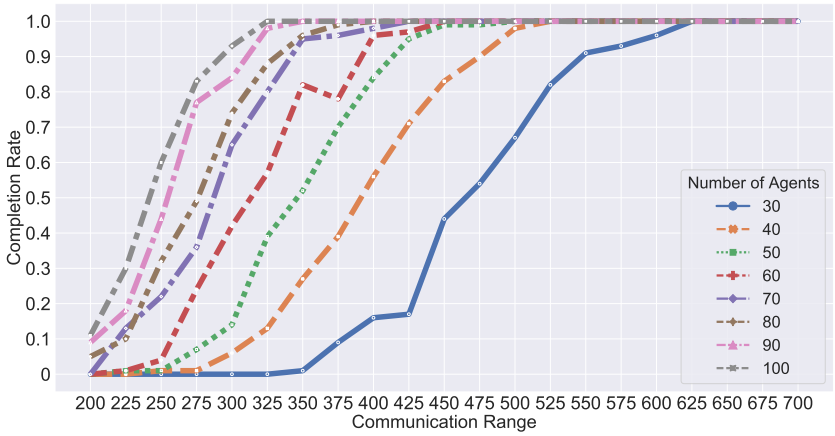
Figure 8 presents the comparison between the proposed method, in Figure 8a, and the original raft-based method, in Figure 8b, with the threshold of 15 votes. The election completion rate of different swarm sizes clearly matches each line graph, see the legends. For example, the blue lines in two subfigures indicate that in a 30-follower swarm, the election completion rate varies with the communication range. As the lowest followers number is 30, we adopt 15 as the vote threshold, exactly equal to half of the swarm. As can be seen, both methods make the completion rate increase with the communication range. Obviously, the proposed method enables a swarm to maintain the lead–follow behavior structure with a relatively smaller size than that of the original Raft-based method. In other words, a swarm using the proposed election mechanism requires about 25% shorter communication range (implying reduced communication condition in real world) to finish a leader election, given the agents number. Moreover, all completion rate lines mostly keep the identical incremental proportion to communication range as each other, which implies that the leader election mechanism is able to scale with the swarm size.

As for the threshold of half swarm size, Figure 9 illustrates the results with a threshold of 50% of the whole follower agents. Overall, the rate for the most part follows the trend as the 15-vote threshold situation, the proposed method showing about 25% better performance than that of Raft-based. Differing from the previous, an interesting phenomenon is that all completion rate lines have nearly the same rising point to communication range. That is because the absolute value of voting threshold increases along with the total number

of agents. The incremental threshold hence demands wider communication area covering more in-swarm individuals to collect adequate votes.

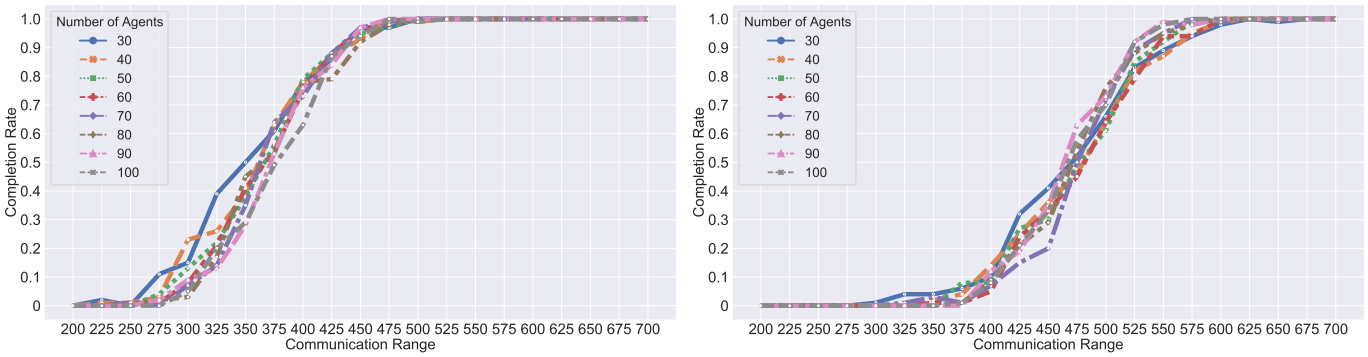



(**a**)




(**b**)

**Figure 8.** Compeletion rate comparison of two methods with election threshold of 15. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.




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

**Figure 9.** Compeletion rate comparison of two methods with election threshold of half of swarm. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.

Furthermore, the comparison of completion rate is illustrated in Figure 10, given 400-unit communication range. It provides deeper insights into performance comparison between the proposed algorithm and the Raft-based method. From Figure 10a, both methods can achieve a high rate under large number circumstances if the threshold is 15. The Raft-based method plunges to a low level in small-sized swarm while our proposed method only faces slight performance reduction. In Figure 10b, it is evident that our proposed method scales smoothly (around 0.7) across different sized swarms; nevertheless, the Raft-based method suffers low completion rates less than 0.1.

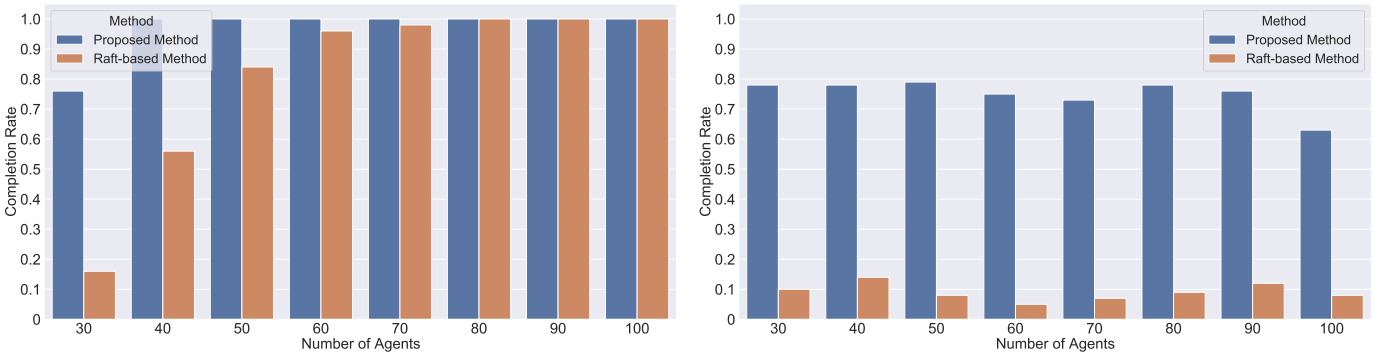



**Figure 10.** The election completion rate over the number of follower agents (the higher the better). (**a**) The comparison between two methods with threshold of 15-vote in 400 units communication range. (**b**) The comparison between two methods with threshold of half swarm size in 400 units communication range.

#### 5.3.2. Success Rate

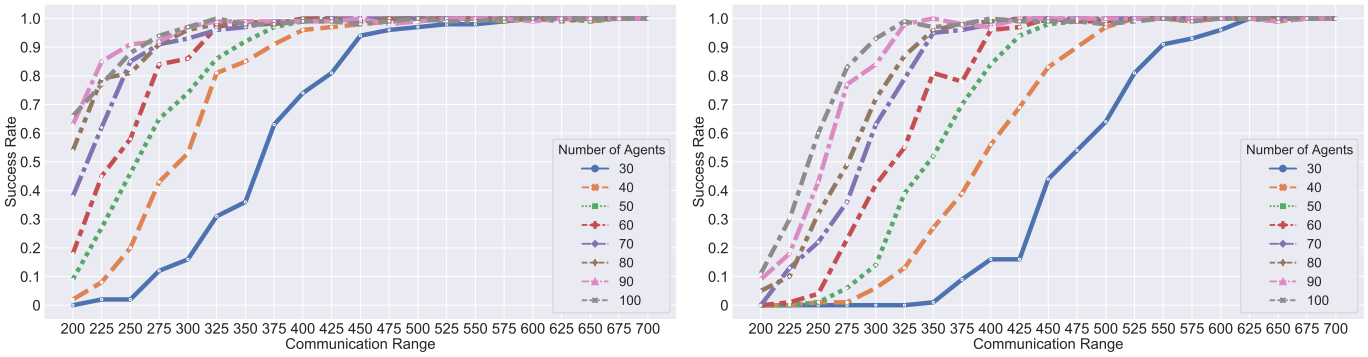
As aforementioned, an agent that has received more votes than a threshold is able to broadcast its winning election. There is a chance that this is any of the *top-n* agents rather than the most eligible one. *Top-n* indicates there are $N$ agents with high qualification measures probably receiving enough votes to end the election process. We take the half-vote threshold of 30 candidates as an example. In theory, an extreme case might take place when the 15th-ranking agent could accept 16 votes and immediately claim lead authority ahead of other top 14 agents. The final eligibility rank of the elected leader would deteriorate, especially when one can cast multiple votes based on the proposed method. The intention of experiments conducted below is to deeply investigate the unexpected effect of vote mechanism imposed on leader election.

Therefore, we define the success of an election as being when the instant eligibility of the elected agent should rank *top-n*. Similar to the completion rate, an election process is considered to be successful if the instant qualification of the elected leader ranks within the *top-n*. In the experiment, *Top-n* is set as *top-10%* and *top-1*, respectively. As a result, the success rate is computed as:

$$SuccRate = \frac{C_{top-n}}{C_{sim}}, \tag{14}$$

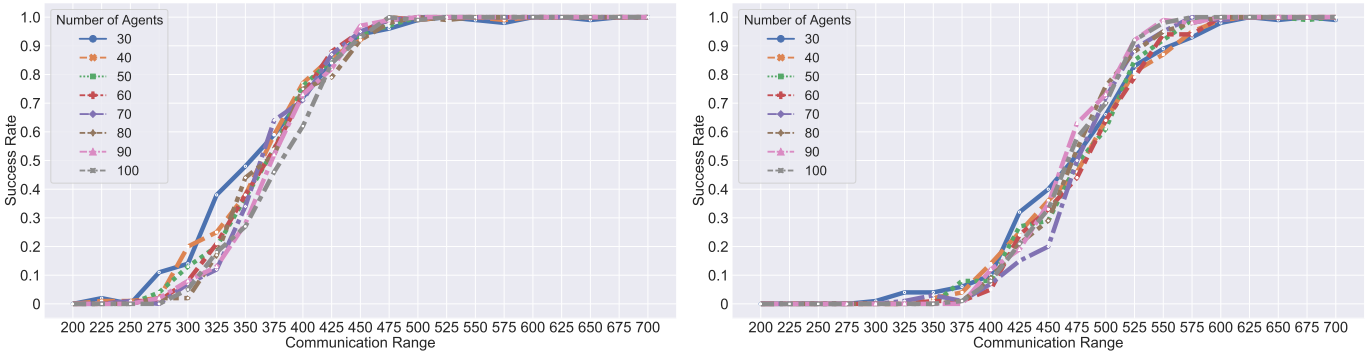
where $C_{top-n}$ is the count of leader agents whose instant qualification measure ranks within top-n among the swarm.

From Figures 11 and 12, the dominant trend of line graphs is consistent with the completion rate. Take the rate value about 0.7–0.8 at y-axis as a case in 12. The swarm is able to reach the point with 400 range thanks to our algorithm, in Figure 12a. On the other hand, the Raft-based method has to shift to 500 range to obtain the same result. Our proposal performs nearly 10–20% better than the original Raft-based method. An significant finding is that although our proposal theoretically suffers higher risks, the success rate still remains proportional to the completion rate in practices.




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

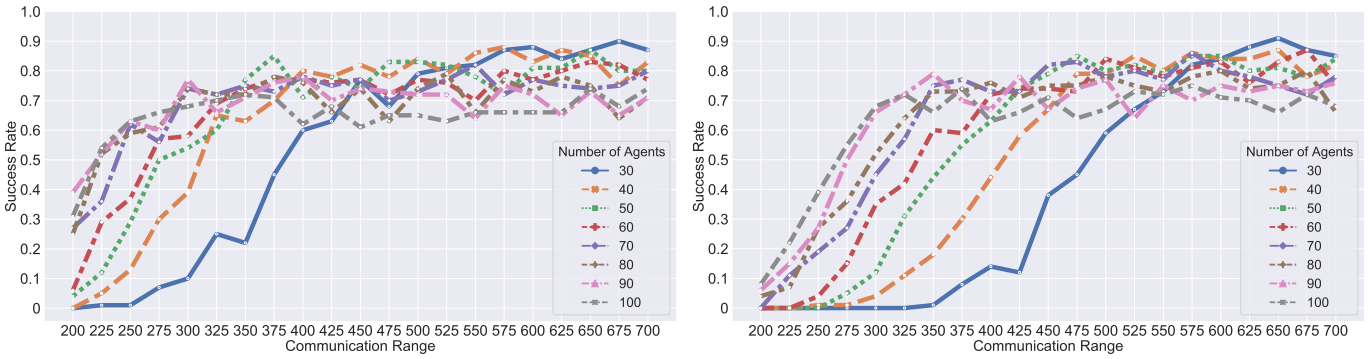
**Figure 11.** The election success rate within top 10 with 15-vote threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.




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

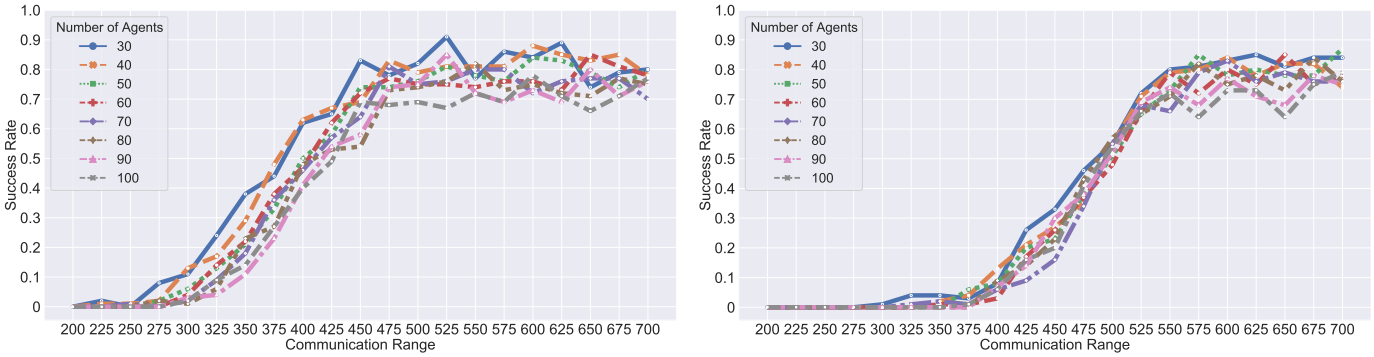
**Figure 12.** The election success rate within top 10 with half swarm size threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.

We further test the election mechansim utilizing a more strict criterion, namely only accounting for the ranking first situation, shown in Figures 13 and 14. The success rate can hold at a high level even under strict constraints (ranking the first), provided the communication requirement is met. The samples that the elected agent ranks at the top occupy the majority of the completion cases. From all the simulated experiments, it is notable that the performance of both election mechanisms can converge to a high level along with increasing communication range. The proposed algorithm in this article obtains about 20% improvement over the Raft-based algorithm.




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

**Figure 13.** The election success rate at top-1 with 15-vote threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.



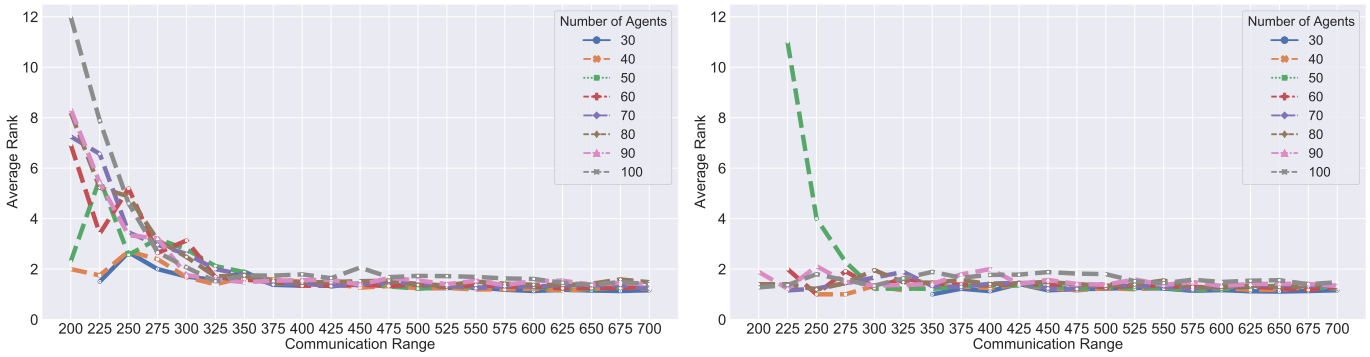

(**a**) (**b**)

**Figure 14.** The election success rate at top-1 with half swarm size threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.

### 5.3.3. Average Rank

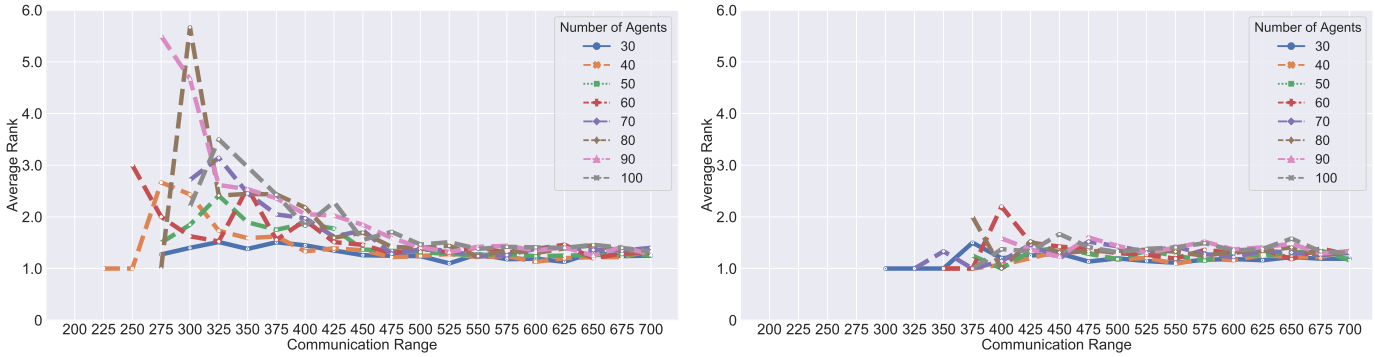
Additionally, we would like to reveal the quality of election in experiments, represented by *average rank*. This follows all basic settings and parameters of the above. The average rank can be regarded as another metric of interest. Here, we mainly focus on the completed simulation samples and omit those failures in order to calculate the average rank.

Same as the above sections, the results of 15-vote and half swarm size threshold are exhibited in Figures 15 and 16, respectively. As can be seen, the rankings fluctuate when the communication range is small while all lines become stable at a slightly higher position than the top with wide ranges. It is understandable because there only exists a small quantity of completion samples, even none below the 300 range in Figure 16a. The completion samples could be less and their ranks are inclined to be low and potentially unpredictable. All four graphs mostly share the identical convergence with the increasing communication range, which inherently coheres with the previous illustration.




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

**Figure 15.** The average rank with 15-vote threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.




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

**Figure 16.** The average rank with half swarm size threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.

#### 5.3.4. Simulation Time

This section aims at unfolding the efficiency of leader election algorithms. That is how long it should take to end the election loop. In the simulation experiments, the election elapsed step is of special interest. We record the election duration from the ending of daemon reconnection loop to the moment that one claims leadership. Since the *Max. Election Elapsed Time* is set as 50 in Table 1, we define a failure as a 50-step sample for the sake of simplicity. That is, if there exists a failure, the time is fixed as 50 steps to unify final results. As such, the less steps that are spent, the better performance is accomplished.

Figures 17 and 18 exhibit simulation time spent in completing election under 15-vote and half swarm size conditions, respectively. Based on the preceding analysis, the election duration time drops to low levels as expected. The simulation steps of large-sized swarms decrease sharply as the communication range widens. From the examinations, our proposed approach achieves less simulated steps than the Raft-based given communication range. It is apparent that the proposal is always able to gain improvements in contrast to the Raft-based way.

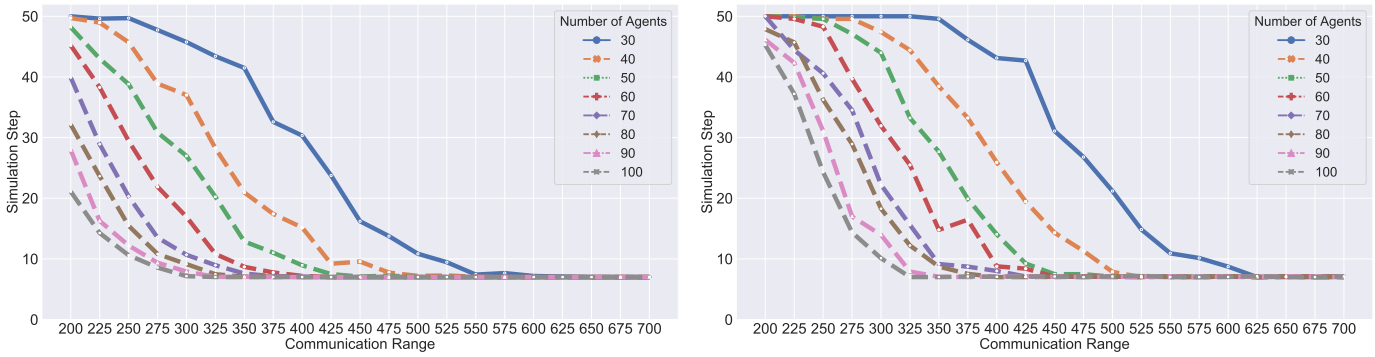



**Figure 17.** The simulation duration with 15-vote threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.

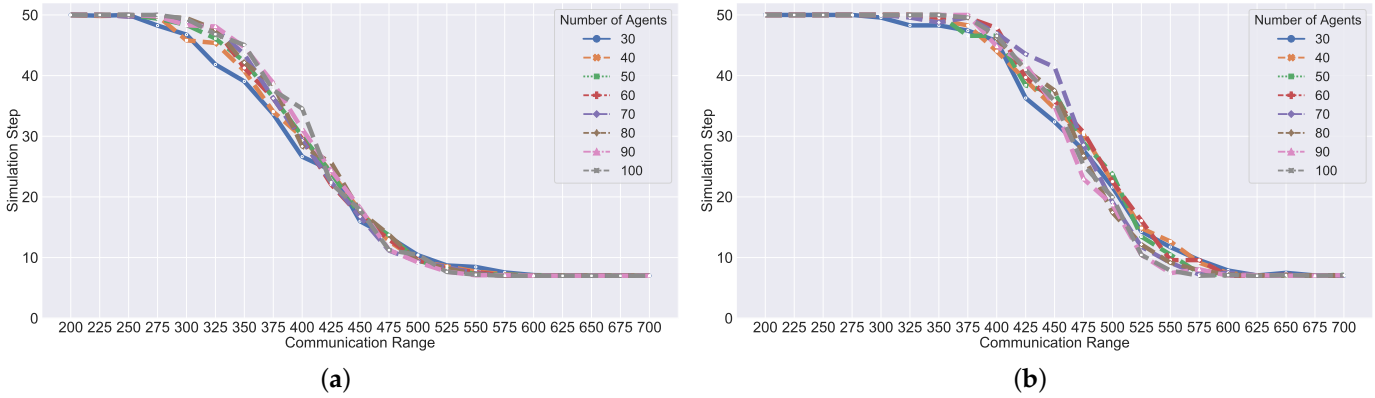



**Figure 18.** The simulation duration with half swarm size threshold. (**a**) Our proposed method with respect to different swarm size. (**b**) The Raft-based method with respect to different swarm size.

#### 5.3.5. Discussion

Another important issue is the energy consumption of UAVs. There are numerous efforts being made to gain energy efficiency and analyze their impact [36]. It is widely believed that there would be two major sources for energy consumption, maneuver [60] and communication [61]. The maneuver of a UAV would consume the majority of its battery compared to onboard computing [60], which implies an approach stimulating task-execution would considerably save energy. On one hand, the experiments illustrate that the proposed method reduces the election duration by around 20% over the original Raft algorithm. It shows a promising potential to expand UAV operation time by decreasing "no task" wandering time and saving battery. On the other hand, the communication conditions, i.e., transmit power, also impose an effect on energy expenditure [36,60]. The simulation examines the performance over communication, reflecting that the UAV energy cost would benefit from our approach owing to the lower communication requirements. In this article, our main concentration is on the voting mode in simulation by distributed and independent interactions. Our future plan is to modify the simulation-based approach to implement in real machines from ground to aerial vehicles, where the actual energy consumption can be accurately measured.

## 6. Conclusions

In this paper, a voting-based leader election scheme is proposed for the fault and disaster recovery of lead–follow formation in UAV and drone swarms. The proposed method is inspired by the Raft algorithm originating from the distributed computation consensus idea. A UAV agent would have three states covering three possible roles: leader,

follower and candidate. Once the original leader encounters a operation failure, the rest would immediately turn to the candidate role and enter into the election process. In the election, the proposal makes each candidate calculate its own metric based on local conditions and only compare itself with the surrounding individuals. The candidate may decide whom it should vote for based on the above comparison. Our proposal allows UAV candidates to *regret* their preceding choice and vote for others including itself based on a dynamic evaluation metric. The metric is utilized as a qualification/eligibility measurement to evaluate to what extent one drone can act as the leader. Four significant principles are additionally listed to help design an elastic measurement for dynamic applications. In this work, the communication condition (denoted as range/radius) is of particular interest. We thereby devise a communication resource pool to constrain communication range. In the end, comprehensive experiments in four aspects illustrate that our proposal is advantageous over the Raft-based scheme and obtains superior performance.

Our future work aims at theoretical combination between formulated consensus and the voting process. Especially, impact of the networking delay on leader election should be deeply discussed in order to discover the potential convergence and limitations. The bandwidth and throughput are also crucial parts of communication conditions, which requires the careful modification for the proposal mechanism in the future work. Moreover, further experiments in terms of communication conditions should be conducted and the proposal requires tests in real flying UAVs to validate the availability and applicability.

**Author Contributions:** Conceptualization, W.Y. and Y.Z. methodology, Y.Z. software, Y.Z.; validation, Y.Z., Q.C. and X.Z.; formal analysis, Y.Z. investigation, J.G. and J.Q.; resources, J.G.; data curation, Y.Z., Q.C. and Y.Z.; writing—original draft preparation, Y.Z. writing—review and editing, Y.Z. and W.Y.; visualization, Y.Z. and J.Q.; supervision, W.Y.; project administration, W.Y.; funding acquisition, J.G. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Science Foundation of China grant No.61902423.

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

## References

1. Mohamed, N.; Al-Jaroodi, J.; Jawhar, I.; Idries, A.; Mohammed, F. Unmanned aerial vehicles applications in future smart cities. *Technol. Forecast. Soc. Chang.* **2020**, *153*, 119293.
2. Ullah, Z.; Al-Turjman, F.; Mostarda, L.; Gagliardi, R. Applications of artificial intelligence and machine learning in smart cities. *Comput. Commun.* **2020**, *154*, 313–323.
3. Wu, D.; Arkhipov, D.I.; Kim, M.; Talcott, C.L.; Regan, A.C.; McCann, J.A.; Venkatasubramanian, N. ADDSEN: Adaptive data processing and dissemination for drone swarms in urban sensing. *IEEE Trans. Comput.* **2016**, *66*, 183–198.
4. Zhang, C. Design and application of fog computing and Internet of Things service platform for smart city. *Future Gener. Comput. Syst.* **2020**, *112*, 630–640.
5. Hu, Z.; Bai, Z.; Yang, Y.; Zheng, Z.; Bian, K.; Song, L. UAV aided aerial-ground IoT for air quality sensing in smart city: Architecture, technologies, and implementation. *IEEE Netw.* **2019**, *33*, 14–22.
6. Neirotti, P.; De Marco, A.; Cagliano, A.C.; Mangano, G.; Scorrano, F. Current trends in Smart City initiatives: Some stylised facts. *Cities* **2014**, *38*, 25–36.
7. Kim, T.h.; Ramos, C.; Mohammed, S. Smart city and IoT. *Future Gener. Comput. Syst.* **2017**, *76*, 159–162.
8. Khan, L.U.; Yaqoob, I.; Tran, N.H.; Kazmi, S.A.; Dang, T.N.; Hong, C.S. Edge-computing-enabled smart cities: A comprehensive survey. *IEEE Internet Things J.* **2020**, *7*, 10200–10232.
9. Wu, Y. Robust learning-enabled intelligence for the internet of things: A survey from the perspectives of noisy data and adversarial examples. *IEEE Internet Things J.* **2020**, *8*, 9568–9579.
10. Jin, J.; Gubbi, J.; Marusic, S.; Palaniswami, M. An information framework for creating a smart city through internet of things. *IEEE Internet Things J.* **2014**, *1*, 112–121.
11. Qadir, Z.; Ullah, F.; Munawar, H.S.; Al-Turjman, F. Addressing disasters in smart cities through UAVs path planning and 5G communications: A systematic review. *Comput. Commun.* **2021**, *168*, 114–135.
12. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1657–1681.
13. Zhao, Z.; Min, G.; Gao, W.; Wu, Y.; Duan, H.; Ni, Q. Deploying edge computing nodes for large-scale IoT: A diversity aware approach. *IEEE Internet Things J.* **2018**, *5*, 3606–3614.

14. Zuo, Y.; Wu, Y.; Min, G.; Huang, C.; Pei, K. An intelligent anomaly detection scheme for micro-services architectures with temporal and spatial data analysis. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *6*, 548–561.
15. Garg, S.; Singh, A.; Batra, S.; Kumar, N.; Yang, L.T. UAV-empowered edge computing environment for cyber-threat detection in smart vehicles. *IEEE Netw.* **2018**, *32*, 42–51.
16. Menouar, H.; Guvenc, I.; Akkaya, K.; Uluagac, A.S.; Kadri, A.; Tuncer, A. UAV-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Commun. Mag.* **2017**, *55*, 22–28.
17. Osseiran, A.; Boccardi, F.; Braun, V.; Kusume, K.; Marsch, P.; Maternia, M.; Queseth, O.; Schellmann, M.; Schotten, H.; Taoka, H.; et al. Scenarios for 5G mobile and wireless communications: The vision of the METIS project. *IEEE Commun. Mag.* **2014**, *52*, 26–35.
18. Xu, Y.; Gui, G.; Gacanin, H.; Adachi, F. A survey on resource allocation for 5G heterogeneous networks: Current research, future trends and challenges. *IEEE Commun. Surv. Tutorials* **2021**, 23, 668–695.
19. Rost, P.; Mannweiler, C.; Michalopoulos, D.S.; Sartori, C.; Sciancalepore, V.; Sastry, N.; Holland, O.; Tayade, S.; Han, B.; Bega, D.; et al. Network slicing to enable scalability and flexibility in 5G mobile networks. *IEEE Commun. Mag.* **2017**, *55*, 72–79.
20. Zuo, Y.; Wu, Y.; Min, G.; Cui, L. Learning-based network path planning for traffic engineering. *Future Gener. Comput. Syst.* **2019**, *92*, 59–67.
21. Shafi, M.; Molisch, A.F.; Smith, P.J.; Haustein, T.; Zhu, P.; De Silva, P.; Tufvesson, F.; Benjebbour, A.; Wunder, G. 5G: A tutorial overview of standards, trials, challenges, deployment, and practice. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 1201–1221.
22. Qi, F.; Zhu, X.; Mang, G.; Kadoch, M.; Li, W. UAV network and IoT in the sky for future smart cities. *IEEE Netw.* **2019**, *33*, 96–101.
23. Xiong, F.; Li, A.; Wang, H.; Tang, L. An SDN-MQTT based communication system for battlefield UAV swarms. *IEEE Commun. Mag.* **2019**, *57*, 41–47.
24. Chen, W.; Liu, B.; Huang, H.; Guo, S.; Zheng, Z. When UAV swarm meets edge-cloud computing: The QoS perspective. *IEEE Netw.* **2019**, *33*, 36–43.
25. Chen, R.; Yang, B.; Zhang, W. Distributed and collaborative localization for swarming UAVs. *IEEE Internet Things J.* **2020**, *8*, 5062–5074.
26. Schranz, M.; Umlauft, M.; Sende, M.; Elmenreich, W. Swarm robotic behaviors and current applications. *Front. Robot.* **2020**, *7*, 36.
27. Brambilla, M.; Ferrante, E.; Birattari, M.; Dorigo, M. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell.* **2013**, *7*, 1–41.
28. Chung, S.J.; Paranjape, A.A.; Dames, P.; Shen, S.; Kumar, V. A survey on aerial swarm robotics. *IEEE Trans. Robot.* **2018**, *34*, 837–855.
29. Wang, H.; Rubenstein, M. Shape formation in homogeneous swarms using local task swapping. *IEEE Trans. Robot.* **2020**, *36*, 597–612.
30. Loria, A.; Dasdemir, J.; Jarquin, N.A. Leader–Follower formation and tracking control of mobile robots along straight paths. *IEEE Trans. Control. Syst. Technol.* **2015**, *24*, 727–732.
31. Ganesan, R.; Raajini, X.M.; Nayyar, A.; Sanjeevikumar, P.; Hossain, E.; Ertas, A.H. BOLD: Bio-inspired optimized leader election for multiple drones. *Sensors* **2020**, *20*, 3134.
32. Cui, J. Kong J Gerla M Zhou S The challenges of building mobile underwater wireless networks for aquatic applications. *IEEE Netw.* **2006**, *20*, 12.
33. Sánchez-García, J.; García-Campos, J.M.; Arzamendia, M.; Reina, D.G.; Toral, S.; Gregor, D. A survey on unmanned aerial and aquatic vehicle multi-hop networks: Wireless communications, evaluation tools and applications. *Comput. Commun.* **2018**, *119*, 43–65.
34. Domingo, M.C. An overview of the internet of underwater things. *J. Netw. Comput. Appl.* **2012**, *35*, 1879–1890.
35. Johansen, T.A.; Zolich, A.; Hansen, T.; Sørensen, A.J. Unmanned aerial vehicle as communication relay for autonomous underwater vehicle—Field tests. In Proceedings of the 2014 IEEE Globecom Workshops (GC Wkshps), Austin, TX, USA, 8–12 December 2014; pp. 1469–1474.
36. 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.
37. Razi, A. Optimal measurement policy for linear measurement systems with applications to UAV network topology prediction. *IEEE Trans. Veh. Technol.* **2019**, *69*, 1970–1981.
38. Razi, A.; Afghah, F.; Chakareski, J. Optimal measurement policy for predicting UAV network topology. In Proceedings of the 51st Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 29 October–1 November 2017; pp. 1374–1378.
39. Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In Proceedings of the USENIX Annual Technical Conference (Usenix ATC 14), Philadelphia, PA, USA, 19–20 June 2014; pp.305–319.
40. Barborak, M.; Dahbura, A.; Malek, M. The consensus problem in fault-tolerant computing. *ACM Comput. Surv. (CSur)* **1993**, *25*, 171–220.
41. Olfati-Saber, R.; Murray, R.M. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Autom. Control.* **2004**, *49*, 1520–1533.
42. Dyer, J.R.; Johansson, A.; Helbing, D.; Couzin, I.D.; Krause, J. Leadership, consensus decision making and collective behaviour in humans. *Philosophical Trans. R. Soc. Biol. Sci.* **2009**, *364*, 781–789.
43. Floreano, D.; Wood, R.J. Science, technology and the future of small autonomous drones. *Nature* **2015**, *521*, 460–466.
44. Gregory, D. From a view to a kill: Drones and late modern war. *Theory Cult. Soc.* **2011**, *28*, 188–215.

45. Zeng, Y.; Wu, Q.; Zhang, R. Accessing from the sky: A tutorial on UAV communications for 5G and beyond. *Proc. IEEE* **2019**, *107*, 2327–2375.
46. Rubenstein, M.; Cornejo, A.; Nagpal, R. Programmable self-assembly in a thousand-robot swarm. *Science* **2014**, *345*, 795–799.
47. Chen, W.; Liu, J.; Guo, H. Achieving robust and efficient consensus for large-scale drone swarm. *IEEE Trans. Veh. Technol.* **2020**, *69*, 15867–15879.
48. Moarref, S.; Kress-Gazit, H. Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications. *Auton. Robot.* **2020**, *44*, 585–600.
49. Kim, H.; Mokdad, L.; Ben-Othman, J. Designing UAV surveillance frameworks for smart city and extensive ocean with differential perspectives. *IEEE Commun. Mag.* **2018**, *56*, 98–104.
50. Zhao, C.; Liu, J.; Sheng, M.; Teng, W.; Zheng, Y.; Li, J. Multi-UAV trajectory planning for energy-efficient content coverage: A decentralized learning-based approach. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3193–3207.
51. Fischer, M.J.; Lynch, N.A.; Paterson, M.S. Impossibility of distributed consensus with one faulty process. *J. ACM (JACM)* **1985**, *32*, 374–382.
52. Gilbert, S.; Lynch, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News* **2002**, *33*, 51–59.
53. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A survey of distributed consensus protocols for blockchain networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1432–1465.
54. Lamport, L. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*; Association for Computing Machinery: New York, NY, USA, October 2019; pp. 277–317. https://doi.org/10.1145/3335772.3335939.
55. Reynolds, C.W. Flocks, Herds and Schools: A distributed behavioral model. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'87), New York, NY, USA, August 1987; pp. 25–34..
56. Mou, Z.; Liu, J.; Yun, X.; Gao, F.; Wu, Q. Cluster Head Detection for Hierarchical UAV Swarm With Graph Self-supervised Learning. *arXiv* **2022**, arXiv:2203.04311.
57. Ma, T.; Zhou, H.; Qian, B.; Fu, A. A large-scale clustering and 3D trajectory optimization approach for UAV swarms. *Sci. China Inf. Sci.* **2021**, *64*, 1–16.
58. Latiff, N.A.; Tsimenidis, C.C.; Sharif, B.S. Energy-aware clustering for wireless sensor networks using particle swarm optimization. In Proceedings of IEEE 18th International Symposium on Personal, Indoor and Mobile rRadio Communications, Athens, Greece, 3–7 September 2007; pp. 1–5.
59. Nuno, E.; Loria, A.; Hernández, T.; Maghenem, M.; Panteley, E. Distributed consensus-formation of force-controlled nonholonomic robots with time-varying delays. *Automatica* **2020**, *120*, 109114.
60. Zhao, Z.; Cumino, P.; Souza, A.; Rosario, D.; Braun, T.; Cerqueira, E.; Gerla, M. Software-defined unmanned aerial vehicles networking for video dissemination services. *Ad Hoc Netw.* **2019**, *83*, 68–77.
61. Zhao, Z.; Dong, W.; Bu, J.; Gu, T., Min, G. Accurate and generic sender selection for bulk data dissemination in low-power wireless networks *IEEE/ACM Trans. Netw.* **2017**, *25*, 948–959.