Computers and Electrical Engineering 99 (2022) 107774

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

A consensus protocol for unmanned aerial vehicle networks in the presence of Byzantine faults☆

Chien-Fu Cheng [a], Gautam Srivastava [b,e], Jerry Chun-Wei Lin [c,*], Ying-Chen Lin [d]

[a] Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung City, Taiwan
[b] Department of Mathematics and Computer Science, Brandon University, Brandon, Canada
[c] Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway
[d] Department of Computer Science and Information Engineering, Tamkang University, New Taipei City, Taiwan
[e] Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan

ARTICLE INFO

Keywords:
Byzantine fault
Consensus problem
Coordination
Fault-tolerant
Unmanned aerial vehicle network

ABSTRACT

This paper discusses the fault-tolerant consensus problem in Unmanned Aerial Vehicle Networks (UAVNets). In recent years, the applications of UAVNets have become more and more popular. Related applications include aerial photography, geological and topographic surveys, disaster monitoring, military applications, and so on. Therefore, it is very important to build a highly reliable and fault-tolerant UAVNets. However, the network architecture of UAVNets is very different from previous network architectures. Because UAVs fly at a high speed, the topology also varies quickly. Hence, how to collect sufficient messages to reach a consensus on the network for UAVs is a challenge. In this paper, the characteristics of the distributed UAVNets will be explored first. Then, based on the characteristics of the UAVNets, a new fault-tolerant consensus protocol called the UAV Consensus Protocol (UCP) is proposed. The proposed UCP consists of two phases: the message exchanging phase and the consensus making phase. The proposed UCP can solve the consensus problem with $\lfloor (n_O-1)/3 \rfloor+1$ rounds of message exchange in the presence of $\lfloor (n_O-1-a_O)/3 \rfloor$ Byzantine faulty UAVs and a_O away UAVs, where n_O is the number of AUVs in the UAVNet. Moreover, the correctness of the proposed UCP is also proved in this paper.

1. Introduction

In recent years, software and hardware technologies have developed rapidly, enabling the application of unmanned aerial vehicle (UAV) systems in the real world [1,2]. Common applications of UAV systems include smart agriculture [3,4], target detection [5,6], search and rescue in post-disaster scenarios [7,8], military surveillance [9,10], etc. In addition, UAVs also play an important role in the data collection of wireless sensor networks (WSNs) [11,12]. Hence, there can be a wide range of applications of UAV systems.

A UAV is composed of a frame, a flight controller, an electronic governor, a motor, a blade, a remote control, and a battery. A UAV network (UAVNet) is composed of groups of UAVs connected via wireless links. When two UAVs are within the communication range, they can directly transmit packets to each other. When the two UAVs are not within the communication range, the packets will be

☆ This paper is for special section VSI-suav. Reviews were processed by Guest Editor Dr. Bhisham Sharma and recommended for publication.
* Corresponding author.
E-mail addresses: jeffcheng@mail.ntou.edu.tw (C.-F. Cheng), srivastavag@brandonu.ca (G. Srivastava), jerrylin@ieee.org (J.C.-W. Lin), 499410438@s99.tku.edu.tw (Y.-C. Lin).

https://doi.org/10.1016/j.compeleceng.2022.107774
Received 5 August 2021; Received in revised form 26 January 2022; Accepted 27 January 2022
Available online 2 February 2022

transmitted to the destination UAV in the network through multi-hop transmission [10,13,14].

Because messages are transmitted wirelessly in UAVNets, they may be vulnerable to malicious attacks. A malicious fault is also known as a Byzantine fault (Definition 1). When Byzantine fault occurs in the UAVNets, UAVs may malfunction, even conspiring with other UAVs infected to send false messages and interfere with the normal operation of the system. The system will then become paralyzed and produce wrong computing results [9,15]. Therefore, it is very important to build a highly reliable and fault-tolerant UAVNet.

Definition 1. Byzantine fault [16]

When Byzantine faults occur, these faulty devices may send incorrect messages, send messages at the wrong time, or do not send messages. In other words, these types of failures may be crash failure, omission failure, transient failure, software failure, temporal failure, etc. Hence, when there are Byzantine faulty devices in the system, it is impossible to predict the activities of these Byzantine faulty devices.

1.1. Related works of the consensus problem

In the field of fault tolerance, solving consensus problems is an important issue [17]. The definition of this problem is as follows. The system contains n devices, and the number of faulty devices is not greater than t, where $n \geq 4$ and $t = \lfloor (n\text{-}1)/3 \rfloor$. For a correct device, it does not know which device is faulty. Communication between devices is done through a direct and reliable link, which means that messages are not manipulated during the transmission. The sender of the message can be identified. Each device has an initial value, the other devices' initial values are obtained utilizing message exchange. After message exchange, the final consensus value for the collected messages can be calculated.

According to the above definition, it can be learned that traditional consensus protocols are for static and fully connected networks. With the emergence of various new types of networks and applications, scholars have been making changes to the definition of the consensus problem to make it conform to the current situation and further address the consensus problem in these new environments and applications. The following are the studies related to the consensus problem in recent years. For example, Alchieri et al. [18] studied the consensus problem with unknown participants through authentic and reliable point-to-point channels. They consider a distributed system composed of participants P drawn from a larger universe U, where $P \subseteqq U$. In a known network, all participants know P, while in an unknown network, a processor i only knows the subset $P_{i,}$ $P_i \subseteqq P$. In [19], Cheng and Huang indicated the existed consensus protocol is inflexible in the handling of initial values. That is, the correct processor cannot change its initial value during the exchange of the message. Therefore, correct processors are very likely to finally agree on values that do not benefit them. For this issue, they studied the harmonized consensus problem in distributed systems. In [20], Cheng and Tsai pointed out that the traditional consensus protocol requires only the computation of the consensus value, regardless of whether that value is good or bad. Furthermore, they also pointed out that existing consensus protocols require that each processor propose its initial value. However, it is not practical to request an initial value from each processor in the real world. Therefore, they proposed a flexible consensus protocol for the consensus problem of alternative plans for distributed systems. The consensus problem is also a core problem in blockchain design. Kuo et al. [21] pointed out that when designing consensus algorithms for blockchain applications, there are two challenges. The first challenge is that each participant should have the same probability of being awarded a prize for his work. The second is that the consensus algorithms must be able to withstand network failures. They formalized the concept of fairness and proposed two consensus protocols to solve the consensus problem for blockchains. The two consensus algorithms are the robust Byzantine agreement (RBA) with strongly fair validity and the hybrid Byzantine agreement (HBA) with both responsiveness and weakly fair validity.

1.2. Motivation

To provide fault-tolerant capability in UAVNets, the consensus problem should be solved in UAVNets. However, the network architecture of UAVNETs is very different from previous network architectures. Firstly, unlike previous 2D environments, UAVNet is a 3D environment. Secondly, UAV flight speeds are very fast, reaching 460 km/h, so the topology changes very quickly [22]. As well as high-speed movement, UAVs may leave and return to the network frequently. Hence, how to collect sufficient messages to reach a consensus on the network for UAVs is a challenge. On the other hand, for UAVs who leave UAVNet, another challenge is to ensure that it can be able to compute a consensus value when it returns. To address the above challenges, this paper will discuss how to solve the consensus problem in UAVNets. The protocol designed for the UAVNets must meet two requirements: (1) Consensus: All correct UAVs compute a common value; (2) Validity: If the initial value of all correct UAVs is v, then all correct UAVs shall agree on v. The main contributions of this paper are as follows:

(1) Each correct UAV can compute a common consensus value: The proposed consensus protocol ensures that all correct UAVs can compute a common consensus value without influence from faulty devices in UAVNets.
(2) The proposed protocol uses to compute a consensus value using a minimum number of exchanges of messages: The detailed description is shown in Section 3.1.3.
(3) The proposed protocol is a distributed protocol: The proposed protocol ensures that each correct UAV can compute a common consensus value in a distributed manner.
(4) The proof of the proposed protocol is provided: The correctness of the proposed protocols is verified.

The remainder of the paper is presented as follows. Detailed information on the formulation of the consensus problem in UAVNets is provided in Section 2. Section 3 provides the approaches of the solution to the consensus problem in UAVNets. The pseudo-code of the approach is presented in Section 4. Section 5 confirms the correctness of the approach. The conclusion and future work are set out in the last section.

2. Formulation of the consensus problem in UAVNets

Fischer et al. [23] proved that the consensus cannot be reached in a strictly asynchronous system. Because of this, the consensus problem is discussed in an asynchronous UAVNet with the partially synchronous assumption [24]. Suppose that there is a set of UAVs O in the UAVNet. Each UAV has its initial value, and the domain range D ={0,1}. Each UAV can be identified with a single unique identity, and each correct UAV does not know the faulty status of other UAVs. Suppose that the type of failure of faulty UAVs is Byzantine fault [16]. This means that the behavior of the faulty UAV is unpredictable. Because all the messages are encrypted during transmission, relay devices cannot falsify messages from senders to receivers. But, Byzantine senders can send inconsistent messages to different UAVs.

$$V(o_i) = V\left(o_j\right), \forall \text{ correctUAV} o_i, o_j \in O \text{ and } i \neq j \tag{1}$$

subject to:

$$I(o_i) \in \{0, 1\}, \forall \text{ UAV } o_i \in O \tag{2}$$

$$n_O \geq 3f_o + a_O + 1 \tag{3}$$

Using the notations in Table 1, the objective function of the proposed protocol is defined as expressed in Eq. (1). Eq. (2) limits the range of the domain of the initial value of each UAV. In addition, to make sure that all correct UAVs can reach a common consensus value, the number of faulty devices should be controlled within a specified amount [25]. In UAVNets, the consensus value is computed by UAVs, so the number of allowed Byzantine faulty UAVs should be constrained. If an excessive number of UAVs exit the network, other UAVs in the network may b ecome unable to acquire sufficient data. To avoid this problem, the number of UAVs allowed to leave the network must also be restricted. Since each UAV periodically sends beacon packets, the number of away UAVs can be known. In summary, the constraint will be shown in Eq. (3). In other words, when the number of Byzantine faulty UAVs is no more than $\lfloor (n_O\text{-}1\text{-}a_O)/3 \rfloor$, the proposed consensus protocol ensures that all correct UAVs can compute a common consensus value without influence from faulty devices in UAVNets. The definition of away UAVs is shown in Definition 2.

Definition 2. away UAV & return UAV

During the execution of consensus protocol, a UAV that flies away from the UAVNet is called an away UAV, while a UAV that returns to the UAVNet before the termination of consensus protocol is called a return UAV.

3. The concept and approach

In the message exchange process of UAVNets, UAVs need to rely on multi-hop transmission to communicate with one another. Because UAVs fly at a high speed, the topology also varies quickly. If UAVs have to temporarily exit the network for certain reasons during the execution of the consensus protocol, these away UAVs will not be able to collect sufficient messages and thus fail to compute a consensus value after returning to the network. Hence, it is necessary to design a consensus protocol for UAVNets that allows UAVs that have temporarily exited from the network during the execution of the algorithm to compute a consensus value after returning to the network. The definition of return UAVs is shown in Definition 2. In this section, the UAV Consensus Protocol (UCP) is proposed to solve the consensus problem in the UAVNet. UCP has two phases, namely the message exchanging phase and consensus making phase.

3.1. Message exchanging phase

The message exchanging phase is designed to collect the messages from other UAVs. The data structure, handling of lost messages, and the number of rounds required of the proposed UCP will be explained in this section. According to message flow, the devices involved in the delivery of messages can be classified into sender, relay, and receiver. To prevent the transmission of messages from being falsified, messages are encrypted [9,15].

Table 1
The notations of the problem formulation.

O ={o_i | 1 ≤i ≤n_o, n_o =|O|}	A set of UAVs, where o_i is an UAV
D ={0,1}	The domain range of the initial value
I(o_i)	The initial value of the UAV o_i
V(o_i)	The consensus value that UAV o_i decides
f_O	The number of allowed Byzantine faulty UAVs
a_O	The number of away UAVs

3.1.1. *Data structure*

During the execution of the proposed UCP, each UAV will store the received messages into a matrix data structure. The matrix is called UCP-matrix. In Fig. 1, the UCP-matrix from the viewpoint of UAVs o_0 is introduced. Assume there are 7 UAVs, respectively denoted by $o_0 \sim o_6$, in the network. When UAV o_0 receives an initial value of another UAV, it will store the initial value in its UCP-matrix $M^1(o_0)$. As shown in the upper left part of Fig. 1, because there are 7 UAVs, there will be 7 elements in $M^1(o_0)$, where value (i) in $M^1(o_0)$ denotes that the value comes from o_i and is stored in M^1 of UAV o_0 (value (i) can also be expressed as $M^1(o_0)[i]$). When UAV o_0 receives the value $M^1(o_i)$ of UAV o_i, it will store the value $M^1(o_i)$ in its UCP-matrix $M^2(o_0)$. As shown in the lower-left part of Fig. 1, because there are 7 elements in its $M^1(o_i)$, there will be 49 elements in $M^2(o_0)$, where value (j,k) in $M^2(o_0)$ denotes that the value is passed from o_j and o_k and is stored in M^2 of UAV o_0 (value (j,k) can also be expressed as $M^2(o_0)[j][k]$). When UAV o_0 receives the value $M^2(o_i)$ of UAV o_i, it will store the value $M^2(o_i)$ in its UCP-matrix $M^3(o_0)$. Given 7 UAVs, UAV o_0 will create 7 matrices including $M^3(o_0)[0]$, $M^3(o_0)[1]$, $M^3(o_0)[2]$, $M^3(o_0)[3]$, $M^3(o_0)[4]$, $M^3(o_0)[5]$, $M^3(o_0)[6]$. The matrix $M^3(o_0)[0]$ is shown in the right of Fig. 1, where value (x,y) in $M^3(o_0)[0]$ implies that the value is passed from o_0 to o_x and then further passed to o_y and is stored in M^3 of UAV o_0 (the value can also be expressed as $M^3(o_0)[0][x][y]$). That is, after the 1st round of message exchange, each o_i will create a matrix of its own $M^1(o_i)$. After the 2nd round of message exchange, each o_i will create a matrix of its own $M^2(o_i)$. This operation continues until the number of rounds of message exchange equals the number of rounds required (the number of rounds required is provided in Section 3.1.3).

3.1.2. *The lost messages from away UAV*

In UAVNets, each UAV can fly freely. Hence, messages from away UAVs cannot be successfully received. The lost messages from away UAVs will be marked by ζ^0. In the following rounds of message exchange, it is necessary to inform other UAVs of UAVs with no value. Our method is as follows: if a UAV receives ζ^j, ζ^{j+1} instead of ζ^j will be stored and used to represent the missing value in the previous round. The purpose of the +1 design is to show how many rounds of message exchange have passed. Since the maximum number of rounds required of the proposed UCP is $\lfloor (n_O\text{-}1)/3 \rfloor+1$, it also implies that $0 \leq j \leq \lfloor (n_O\text{-}1\text{-}a_O)/3 \rfloor$. Details about the maximum number of rounds required of the proposed UCP are mentioned in the next section. By doing so, it can tell if a value is an unsent value of earlier UAVs or an unsent value of the current UAV.

3.1.3. *The number of rounds required of UCP*

The term round is used to compute the amount of message exchange. A round of message exchange comprises three steps as follows: (1) send messages to any receiver; (2) receive messages from others, and (3) conduct local processing [25]. Moreover, Fischer and Lynch [25] point out that given a network consisting of some Byzantine faulty processors, if all the correct processors are unable to know which processor is faulty, and the number of Byzantine faulty processors is smaller than or equal to t ($t = \lfloor (n\text{-}1)/3 \rfloor$, where n is the number of processors in the network), these correct processors can compute a consensus value after $t+1$ rounds of message exchange. Our assumption for the failure type of fallible processors (i.e. UAVs) and the definition of a round is the same as those used in Fischer and Lynch [25]. Therefore, the maximum number of rounds required of the proposed UCP is $\lfloor (n_O\text{-}1)/3 \rfloor+1$, where n_O is the number of UAVs in the UAVNet.

3.2. *Consensus making phase*

In this section, (1)how to calculate the consensus value for correct UAVs which have never exited the UAVNet (i.e. non-away UAVs) and (2)how to return UAVs can obtain a consistent consensus value from other UAVs after returning the UAVNet are explained.

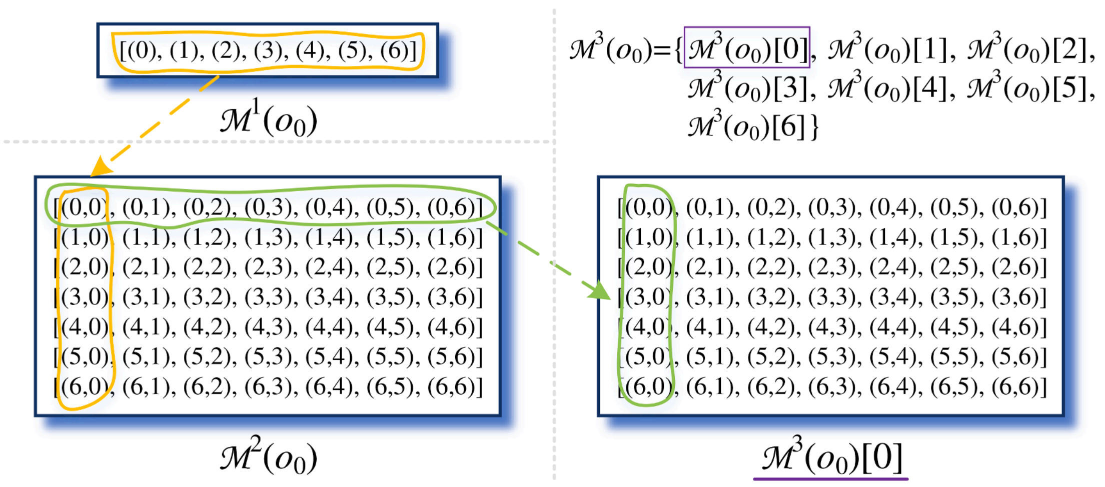

Fig. 1. Examples of UCP-matrix.

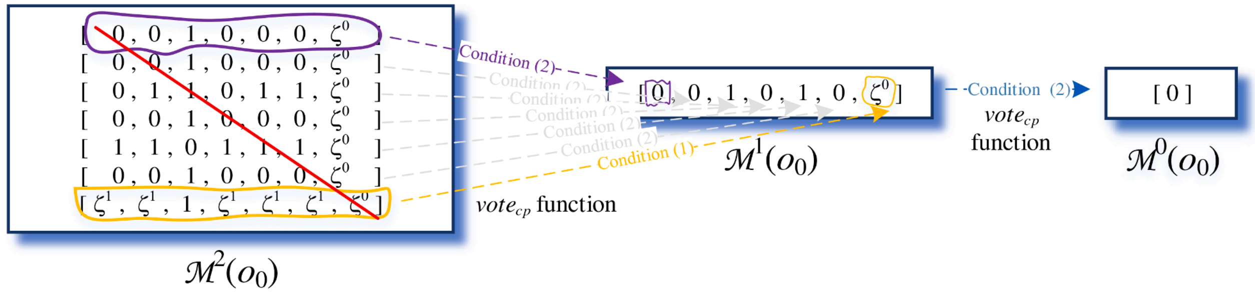
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[0 , 1 , 1 , 0 , 1 , 1 , ζ^0]
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[1 , 1 , 0 , 1 , 1 , 1 , ζ^0]
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[ζ^1 , ζ^1 , 1 , ζ^1 , ζ^1 , ζ^1 , ζ^0]
$\mathcal{M}^2(o_0)$
Condition (2)
Condition (1)
$vote_{cp}$ function
[0 , 0 , 1 , 0 , 1 , 0 , ζ^0]
$\mathcal{M}^1(o_0)$
Condition (2)
$vote_{cp}$ function
[0]
$\mathcal{M}^0(o_0)$

Fig. 2. An example of executing *vote*$_{cp}$ function.

3.2.1. Calculating the consensus value for non-away UAVs

After $\llcorner(n_O\text{-}1)/3\lrcorner+1$ rounds of message exchange, each correct UAV o_i that did not fly away from the UAVNet during the message exchanging phase will create a matrix of its own UCP-matrix $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor+1}(o_i)$. In the message exchange process, certain messages are repeatedly transmitted by the same UAV. To avoid repeated interference, the values transmitted to the same UAVs are deleted. For example, $M^3(O_i)[3][2][2]$ (i.e. element [3][2][2] in $M^3(O_i)$) denotes a value that is passed from o_3 to o_2 and then further passed to o_2 again. The value of $M^3(O_i)[3][2][2]$ is deleted. Later, it converts $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor+1}$ into $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor}$ using a majority function and then converts $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor}$ into $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor-1}$. This conversion process continues until M^0, which is the consensus value. It should be noted that the impact of lost messages should be considered in the design of the majority function. The proposed majority function is called $vote_{matrix}$. The $vote_{matrix}$ function only counts the non-value ζ° for all elements in UCP-matrix. There are three conditions in the $vote_{matrix}$ function. Condition (1): if the majority value is ζ^j then output the value ζ^{j-1}, where $1 \leq j \leq \llcorner(n_O\text{-}1)/3\lrcorner$. Condition (2): if the majority value is non-ζ^j value then output the majority value m, where $m\in\{0,1\}$. Condition (3): if the majority does not exist then output the default value ϕ. Take Figure 2 as an example. Using the $vote_{matrix}$ function, it can convert $M^2(o_0)$ into $M^1(o_0)$, where $M^1(o_0)[0]$ comes from the $vote_{matrix}$ function value of row 0 in $M^2(o_0)$, $M^1(o_0)[1]$ comes from the $vote_{matrix}$ function value of row 1 in $M^2(o_0)$, and so on. Later, it uses the $vote_{matrix}$ function to convert $M^1(o_0)$ into $M^0(o_0)$, where $M^0(o_0)[0]$ comes from the $vote_{matrix}$ function value of row 0 in $M^1(o_0)$. The value of $M^0(o_i)$ is its consensus value.

3.2.2. Calculating the consensus value for return UAVs

When a return UAV returns to the UAVNet, if it is a correct UAV, it has to ensure that the consensus value it gets matches the value obtained by other correct UAVs that have never gone away. Our method is that the return UAV will request the consensus values from UAVs that have never been away. When $\llcorner(n_O\text{-}a_O\text{-}1)/3\lrcorner+1$ uniform values are collected, it will set this uniform value, denoted by V, as its consensus value. Why is a value repeated $\llcorner(n_O\text{-}a_O\text{-}1)/3\lrcorner+1$ times can be the consensus value for this return UAV? This is because an UAVNet has a maximum number of $\llcorner(n_O\text{-}a_O\text{-}1)/3\lrcorner$ Byzantine faulty UAVs (see Eq. (3)), and if the return UAV can collect the same value $\llcorner(n_O\text{-}a_O\text{-}1)/3\lrcorner+1$ times, this value must come from correct UAVs.

4. The proposed protocol

In this section, pseudo-code is used to explain how UCP works. According to the functions of Table 2, the pseudo-code of the proposed UCP is presented in Algorithm 1. In the initial stage, each UAV o_i will store its initial value $I(o_i)$ in its UCP-matrix $M^0(o_i)$ (Line 1 in the UCP) and then empty the UCP-matrix (Line 2,3 in the UCP).

Later, the UAVs enter the message exchange phase (Line 4–11 in the UCP). The number of rounds needed at the phase of message exchange is $\llcorner(n_O\text{-}1)/3\lrcorner+1$ (Line 4 in the UCP). In each round of message exchange, each UAV o_i will send the UCP-matrix it has received in the previous round to all UAVs (Line 5,6 in the UCP). For example, UAV o_i will send $M^0(o_i)$ in round 1, $M^1(o_i)$ in round 2, …, and $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor}(o_i)$ in the last round, which is round $\llcorner(n_O\text{-}1)/3\lrcorner+1$. In each round, each UAV o_i will take some time to collect the UCP-matrix from other UAVs. The duration of this period can be determined by the user depending on the status of the network (Line 7–10 in the UCP). In an UAVNet, UAVs may exit the network temporarily or perpetually during the execution of VAP. Hence, a portion of messages from these UAVs will not be collected. These missing messages are marked (Line 11 in the UCP).

The operation of the consensus making phase is divided into two cases: Case 1: UAV o_i has been away during execution of UCP (Line 12–16 in the UCP) and Case 2: UAV o_i has never been away during execution of UCP (Line 17–25 in the UCP). In Case 1, UAV o_i can use the UCP-matrix $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor+1}(o_i)$ to compute the consensus value (Line 12–16). To avoid repetitive interference in the message exchange process, values that are passed to the same UAVs are deleted (Line 13,14 in the UCP). Later, it can use the $vote_{cp}$ function to progressively convert UCP-matrix $\boldsymbol{M}^{\lfloor(n_O-1)/3\rfloor+1}(o_i)$ into $M^0(o_i)$. This removes the influence of Byzantine faulty UAVs and away UAVs (Line 15,16 in the UCP). In Case 2, UAV o_i has to empty its UCP-matrix $M^1(o_i)$ and wait for other UAVs that have never been away to complete computing of the consensus value (Line 18,19 in the UCP). Subsequently, UAV o_i will request the consensus value from these

Table 2
The functions used in the proposed UCP.

$M^h(o_i)$	$M^h(o_i)$ is where o_i stores the data its obtains during message exchange. M^h denotes a h-dimension array. When $h = 0$, it is a variable. $M^0(o_i)$ stores the initial value of o_i; $M^1(o_i)$ stores the values that o_i has collected in the 1st round of message exchange; $M^h(o_i)$ stores the values that o_i has collected in the h round of message exchange.
$snd(\langle mg\rangle, rcv)$	send a message $\langle mg\rangle$ using the encryption technology [9] to receiver rcv. $snd(\langle \text{Exh}, M^{r\text{-}1}(o_i), o_j, r\rangle, o_j)$: send an Exh message which destination is o_j with the matrix $M^{r\text{-}1}(o_i)$ and round r to receiver o_j. $snd(\langle \text{Fnl}, V(o_i)\rangle, o_j)$: send a Fnl message with the value $V(o_i)$ to receiver o_j. $snd(\langle \text{Req}\rangle, o_j)$: send a Req message to receiver o_j.
$absent(M^h(o_i))$	$absent(M^h(o_i))$ is used to check if there is any value related to ζ^j in $M^h(o_i)$. If positive, change the value to ζ^{j+1}.
$del(M^h(o_i))$	$del(M^h(o_i))$ is used to remove values with the same index value in any two dimensions. For example, if the first-dimension index value of $M^5(o_i)[1][2][3][4][1]$ is the same as the fifth-dimension index value of $M^5(o_i)[1][2][3][4][1]$, the value in $M^5(o_i)[1][2][3][4][1]$ will be deleted.
$vote_{cp}(M^h(o_i))$	According to the matrix structure, $vote_{cp}(M^h(o_i))$ is used to convert h-dimension matrix into h-1-dimension ($M^h(O_i)$ into $M^{h\text{-}1}(O_i)$) (Section 3.2.1). Finally, the majority values are restored in the corresponding locations in $M^{h\text{-}1}(o_i)$.
$suff(M^1(o_i))$	if $\lvert M^1(o_i)\rvert_v = \llcorner(n_O\text{-}a_O\text{-}1)/3\lrcorner+1$, $suff(M^1(o_i))$=*true*; otherwise $suff(M^1(o_i))$=*false*, where v=$maj(M^1(O_i))$.
$maj(M^1(o_i))$	$maj(M^1(o_i))$ is used to exclude ζ value and extract the majority value in matrix $M^1(o_i)$.
$return(o_i)$	if o_i is a return UAV, $return(o_i)$=*true*; otherwise, $return(o_i)$=*false*.

UAVs until the number of the majority value in $M^1(o_i)$ reaches $\lfloor(n_O\text{-}a_O\text{-}1)/3\rfloor+1$ (Line 21–24 in the UCP). Finally, it can also use the $vote_{cp}$ function to progressively convert UCP-matrix $M^1(o_i)$ into $M^0(o_i)$. The value of $M^0(o_i)$ is its consensus value (Line 26 in the UCP).

Algorithm 1. The proposed UCP

```
Algorithm: UAV Consensus Protocol (UCP) //for each UAV $o_i$
Input: $I(o_i)$ // $I(o_i)$ is the initial value of UAV $o_i$
Output: V // V is the consensus value
/* Initialization
1. $M^0(o_i) \leftarrow I(o_i)$;
2. for ($j=1$; $j \le \lfloor(n_O\text{-}1)/3\rfloor+1$; $j$++) do
3.    $M^j(o_i) \leftarrow \zeta^{-1}$;
/* Message Exchanging Phase
4. for ($r=1$; $r \le \lfloor(n_O\text{-}1)/3\rfloor+1$; $r$++) do
5.    for ($j=0$; $j \le n_O\text{-}1$; $j$++) do
6.       snd(⟨Exh, $M^{r\text{-}1}(o_i)$, $o_j$, $r$⟩, $o_j$);
7.       $T \leftarrow c + \eta$;
8.    while ($c < T$) do
9.       rcv&proc();
10.        $T \leftarrow -\infty$;
11.   absent($M^r(o_i)$);
/* Consensus Making Phase
12. If return($o_i$)=false then
13.   for ($j=\lfloor(n_O\text{-}1)/3\rfloor+1$; $j \ge 1$; $j$–)
14.      del($M^j(o_i)$);
15.   for ($j=\lfloor(n_O\text{-}1)/3\rfloor$; $j \ge 1$; $j$–)
16.      $M^{j\text{-}1}(o_i) \leftarrow vote_{cp}(M^j(o_i))$;
17. else
18.   $M^1(o_i) \leftarrow \zeta^{-1}$;
19.   wait until ($c > E$) do
20.      for $o_j \in O$ do
21.         while (suff($M^1(o_i)$=false) do
22.            if return($o_j$)=false then
23.               snd(⟨Req⟩, $o_j$);
24.                  rcv&proc();
25.   $M^0(o_i) \leftarrow vote_{cp}(M^1(o_i))$;
26. return $M^0(o_i)$;
/* Function rcv&proc()
27. function rcv&proc()
28.   for each ($o_j$, ⟨Exh, $M^{r\text{-}1}(o_j)$, $o_i$, $r$⟩) do
29.      $M^r(o_i)[c_1]\ldots[c_{r\text{-}1}][j] \leftarrow M^{r\text{-}1}(o_j)$;
30.   for each ($o_j$, ⟨Fnl, V($o_j$)⟩) do
31.      $M^1(o_i)[j] \leftarrow V(o_j)$;
```

ü $M^h(o_i)$: $M^h(o_i)$ is where o_i stores the data its obtains during message exchange. M^h is a h-dimension array. When $h=0$, it is a variable. $M^0(o_i)$ stores the initial value of o_i; $M^1(o_i)$ stores the values that o_i has collected in the 1st round of message exchange; $M^h(o_i)$ stores the values that o_i has collected in the h round of message exchange.
ü c: current time
ü η: the waiting time allowed for one round
ü T: a point in time
ü E: the finish time when the non-away UAV obtains the consensus value

4.1. An example

The proposed UCP can be applied to many applications of UAVNets. For example, the leader-election issue is an important topic of the UAVNets. The proposed UCP can be used to elect a group leader in the presence of the Byzantine faulty UAVs. In the following, an example is used to explain the operation of the proposed UCP. The setting of this example is as follows: An UAVNet consists of seven UAVs, and each UAV has an initial value. Suppose one of the UAVs will fly away from the UAVNet during the execution of the proposed UCP. According to the constraint of Eq. (3), the number of allowed Byzantine faulty UAVs in this example is one ($\lfloor(n_O\text{-}1\text{-}a_O)/3\rfloor = \lfloor(7\text{–}1\text{-}1)/3\rfloor = 1$). Assume that UAV o_2 is the Byzantine faulty UAV of these UAVs. The initial values of all correct UAVs are shown in Table 3. An example of UAVNet is shown in Fig. 3. During the execution of the proposed UCP, UAV o_6 will fly away from the UAVNet. The operation of the UCP from the perspective of UAV o_0 is explained.

The number of message exchange rounds required of this example is 3 ($\lfloor(n_O\text{-}1)/3\rfloor+1 = \lfloor(7\text{–}1)/3\rfloor+1 = 3$). In the 1st round of message exchange, o_0 will send its initial value to all other UAVs and receive the initial values from other UAVs in the network by encryption technology [9]. Subsequently, UAV o_0 will store the initial values received from other UAVs in its UCP-matrix $M^1(o_0)$. The UCP-matrix M^1 constructed by each correct UAV is shown in Fig. 4. Because UAV o_2 is a Byzantine faulty UAV, o_2 will send inconsistent initial values to other UAVs in the network to disturb correct UAVs. In other words, the values in $M^1(o_0)[2]$, $M^1(o_1)[2]$, $M^1(o_3)[2]$, $M^1(o_4)[2]$, $M^1(o_5)[2]$, and $M^1(o_6)[2]$ will not be consistent. To better understand the operation of UCP, the message through o_2 is marked in red. In the 2nd round of the message exchanging phase, o_6 flies away from the UAVNet. Therefore, UAVs in the network cannot receive the message from o_6. The value ζ^0 is used to report an omitted message. The UCP-matrix $M^2(o_0)$ constructed by o_0 in the 2nd round of the message exchanging phase is shown in Fig. 5. In the last round of message exchange (i.e. the 3rd round of message exchange), o_0 will send its $M^2(o_0)$ to all other UAVs and receive the M^2 from other UAVs. The UCP-matrix M^3 constructed by o_0 is shown in Fig. 6.

After three rounds of message exchange, UAV o_i will enter the consensus-making phase. As shown in the left part of Fig. 7, o_i will delete values delivered to the same UAVs to exclude repetitive influence and then use the $vote_{cp}$ function to convert UCP-matrix $M^3(o_0)$ into $M^2(o_0)$, $M^2(o_0)$ into $M^1(o_0)$, and finally $M^1(o_0)$ into $M^0(o_0)$. By doing so, it can exclude the influence from Byzantine faulty UAVs

Table 3
The initial values of all correct UAVs.

	o_0	o_1	o_3	o_4	o_5	o_6
Initial value	0	0	0	1	0	1

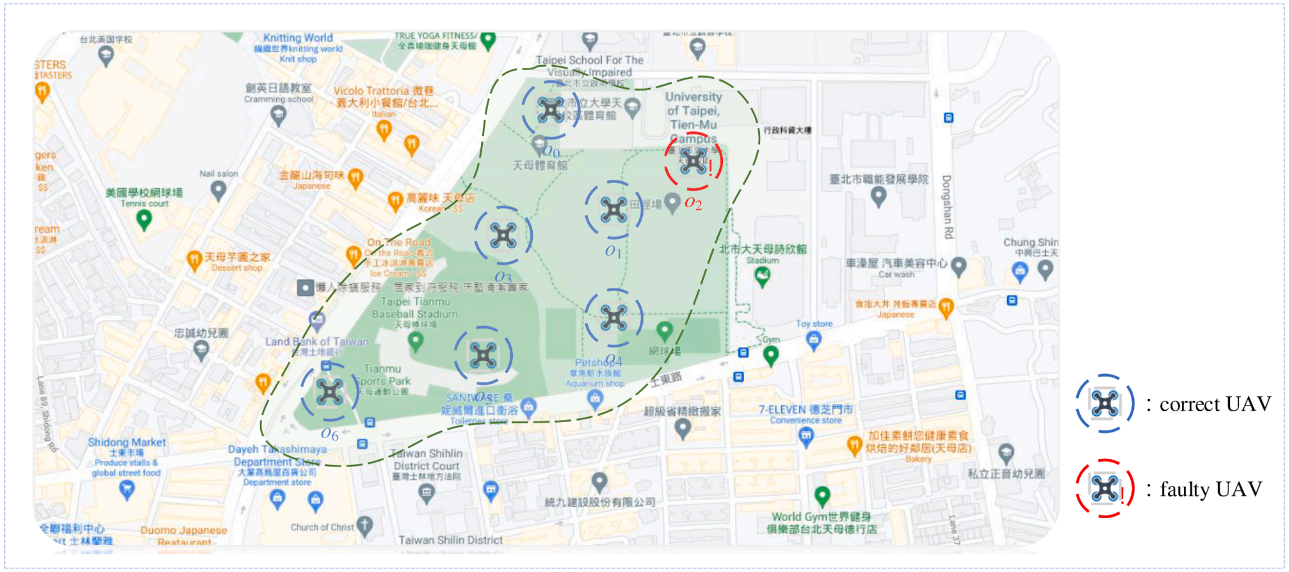

Fig. 3. An example of UAVNet.

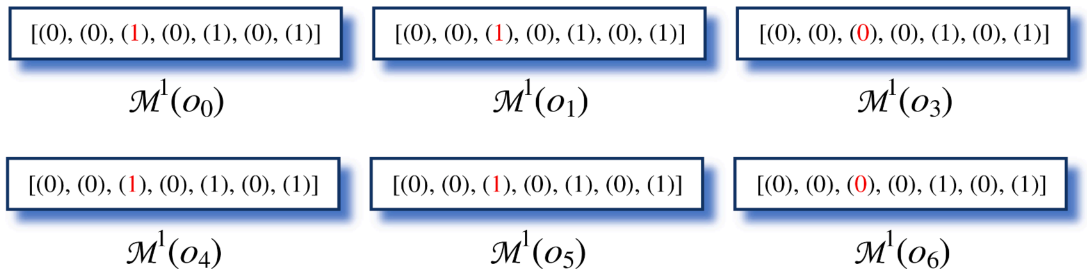

Fig. 4. The UCP-matrix M^1 of all correct UAVs after the 1st round of message exchanging phase.

and obtain a consensus value (as shown in the right of Fig. 7). The consensus value is 0 in this example.

5. The correctness

The correctness of the proposed UCP can be proved if all the correct UAVs have the same UCP-matrix M^0. An element [σ] is called the same if the value stored at element [σ] of all correct UAVs' UCP-matrix are the same. The following two terms are defined to prove that a consensus value can be computed by UCP: (1) Correct element: element $[\alpha][i]$ of a UCP-matrix is eligible as a correct element if UAV o_i (the last UAV id in element $[\alpha][i]$, where α is a sequence of UAV id and i is a single UAV id) is correct. That is, a correct element is a place to store the value received from a correct UAV. (2) True value: For a correct element $[\alpha][i]$ in the UCP-matrix of a correct UAV o_j, $val([\alpha][i])$ is the true value of element $[\alpha][i]$. That is, the stored value of the correct element $[\alpha][i]$ is called the true value if the UAV o_j is correct.

Lemma 1. ***All correct elements of a UCP-matrix are the same.***

Proof: In $\boldsymbol{M}^{\lfloor (n_O-1)/3 \rfloor+1}$ or above UCP-matrices, the correct element has at least $2f_o+1$ children, out of which at least f_o are correct. The true values of these f_o+1 correct elements are the same, and the majority value of elements is the same. The correct element is the same in the UCP-matrix if the level is lower than f_o+1. As a result, all correct elements of the UCP-matrix are the same.

Lemma 2. ***The same frontier exists in the UCP-matrices.***

Proof: There are f_o+1 elements along each $\boldsymbol{M}^0$-to-$\boldsymbol{M}^{\lfloor (n_O-1)/3 \rfloor+1}$ path of UCP-matrices. Since at most f_o Byzantine faulty UAVs can be failed, there is at least one correct element along each $\boldsymbol{M}^0$-to-$\boldsymbol{M}^{\lfloor (n_O-1)/3 \rfloor+1}$ path of UCP-matrices. By Lemma 1, the correct element is the same, and the same frontier exists in each correct UAV's UCP-matrices.

Lemma 3. ***Let [α] be an element; [α] is the same if there is the same frontier in the sub-UCP-matrices rooted at [α].***

Proof: The height of $[\alpha]$ in M^i is i. If the height of $[\alpha]$ is 0 and the same frontier ($[\alpha]$ itself) exists, then $[\alpha]$ is the same. If the height of $[\alpha]$ is r, the children of $[\alpha]$ are all the same under the induction hypothesis with the height of the children being r-1.

Corollary 1. ***The UCP-matrix M^0 is the same if the same frontier exists in the UCP-matrices.***

Theorem 1. ***The UCP-matrix M^0 of a correct UAV's UCP-matrices is the same.***

Proof: By Lemmas 1–3, and Corollary 1, the theorem is proved.

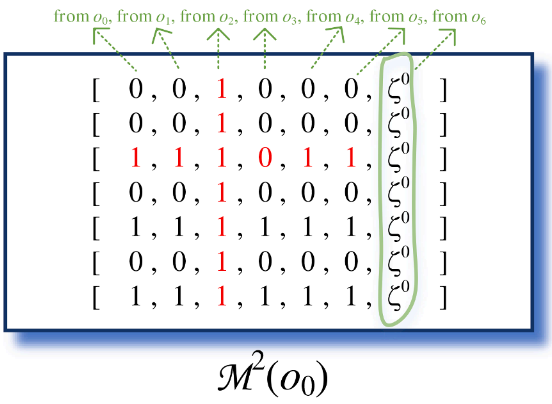
from o_0, from o_1, from o_2, from o_3, from o_4, from o_5, from o_6
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[1 , 1 , 1 , 0 , 1 , 1 , ζ^0]
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[1 , 1 , 1 , 1 , 1 , 1 , ζ^0]
[0 , 0 , 1 , 0 , 0 , 0 , ζ^0]
[1 , 1 , 1 , 1 , 1 , 1 , ζ^0]
$\mathcal{M}^2(o_0)$

Fig. 5. The UCP-matrix M^2 of UAV o_0 after the 2nd round of message exchanging phase.

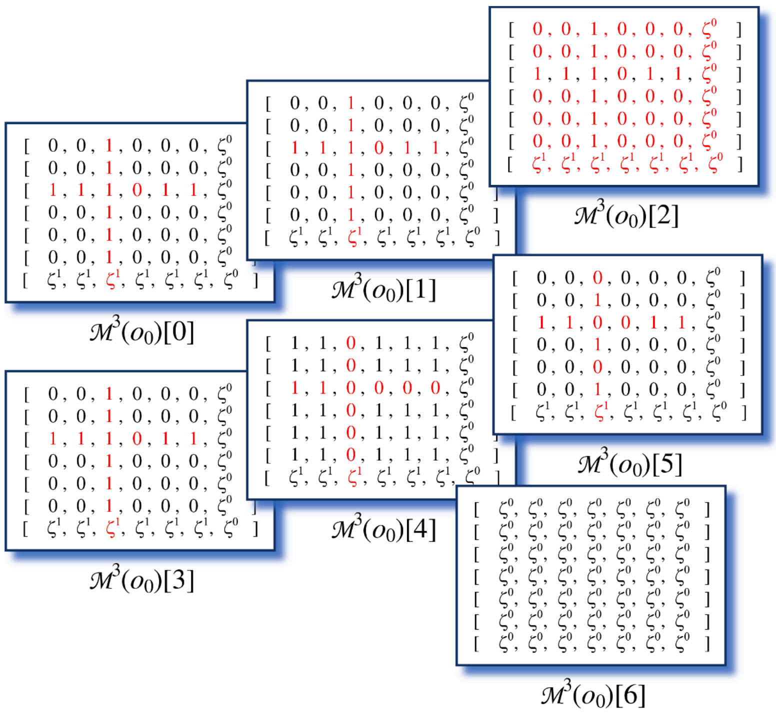

Fig. 6. The UCP-matrix M^3 of UAV o_0 after the 3rd round of message exchanging phase.

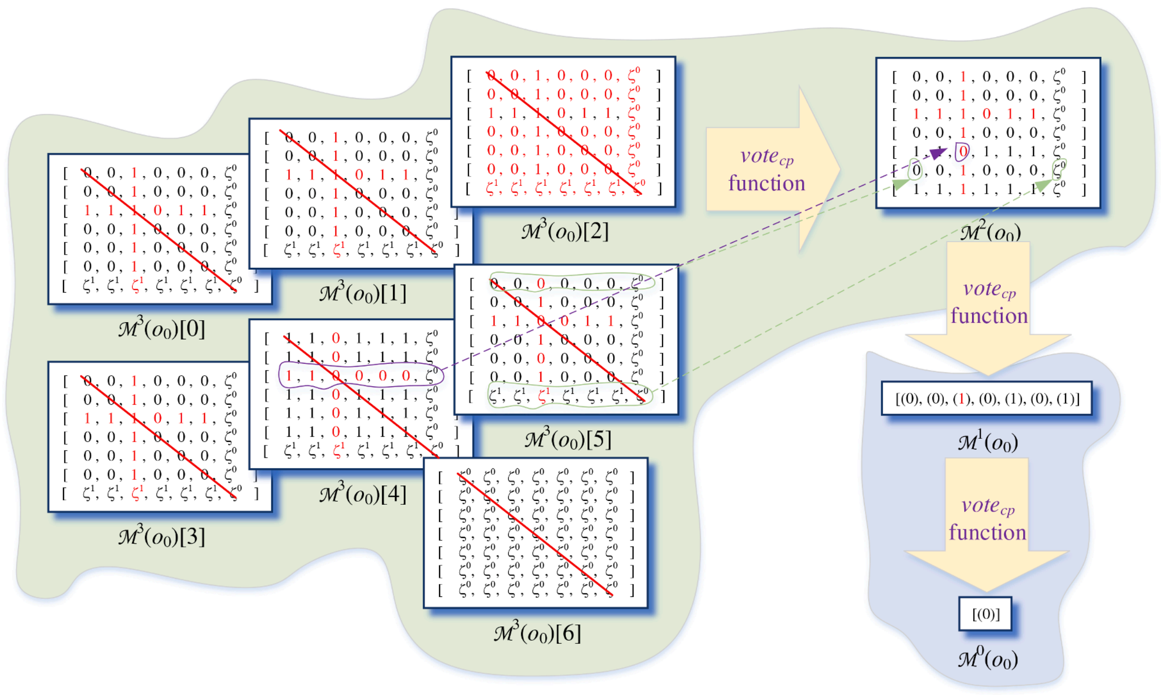

Fig. 7. An example of computing the consensus value.

Theorem 2. ***UCP can solve the consensus problem in an UAVNet.***

Proof: UCP must meet the constraints (Consensus') and (Validity') to prove the theorem.

(Consensus'):	The value of M^0 is the same.

(continued on next page)

(*continued*)

	By Theorem 1, (Consensus') is satisfied.
(Validity'):	$vote_{cp}(M^0) = v$ for all correct UAVs, if the initial value of each correct UAV is v_i say $v = v_i$ The value of the correct elements for the UCP-matrices of all the correct UAVs is v. As a result, each correct element of the UCP-matrices is the same (Lemma 2), and its true value is v. By Theorem 1, the M^0 is the same. The computed value $vote_{cp}(M^0)= v$ is stored in the M^0 for all the correct UAVs. Therefore, (Validity') is satisfied.

6. Conclusion

The development of UAV systems can provide us with a more efficient and convenient life. Hence, fault tolerance of UAVNets is of high importance, and it is necessary to investigate the consensus problem in UAVNets. However, UAVs have the characteristic of high-speed movement, which creates the situation that topology changes quickly and UAVs frequently enter and leave the network. In this case, UAVs away from networks are unable to collect enough messages to reach a consensus. In addition, UAVs that have not left the network before will not receive messages from those away UAVs. This is why the previous consensus protocols do not apply to the current UAVNets. To deal with the characteristics of UAVNets mentioned above, a new consensus protocol UCP is proposed. For missing messages of away UAVs, the proposed UCP marked these messages in the UCP-matrix. When the away UAVs return to the network, they request the consensus value from other UAVs that have never been away. Through the $vote_{cp}$ function of UCP, all the correct UAVs can compute a consensus value in UAVNets without influence from Byzantine faulty UAVs if the number of Byzantine faulty UAVs is smaller than $\lfloor(n_O\text{-}1\text{-}a_O)/3\rfloor$, where a_O is the number of away UAVs, n_O is the number of AUVs. This paper addresses the problem of fault tolerance in UAVNets, but to provide more reliable UAVNets, it is also necessary to detect and locate faulty UAVs. Therefore, it is also necessary to design a detection mechanism to find faulty UAVs. Afterward, these faulty UAVs can be isolated. This is the direction of future research.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Reference

[1] Garcia Sanchez S, Mohanti S, Jaisinghani D, Chowdhury KR. Millimeter-wave base stations in the sky: an experimental study of UAV-to-ground communications. IEEE Trans Mob Comput 2021. https://doi.org/10.1109/TMC.2020.3013575.
[2] Silic M, Mohseni K. Correcting current-induced magnetometer errors on UAVs: an online model-based approach. IEEE Sens J 2020;20(2):1067–76.
[3] Anand T, Sinha S, Mandal M, Chamola V, Richard Yu F. AgriSegNet: deep aerial semantic segmentation framework for IoT-assisted precision agriculture. IEEE Sens J 2021;21(14):17581–90.
[4] Maddikunta PKR, Hakak S, Alazab M, Bhattacharya S, Gadekallu TR, Khan WZ, Pham Q. Unmanned aerial vehicles in smart agriculture: applications, requirements, and challenges. IEEE Sens J 2021;21(16):17608–19.
[5] Bouguettaya A, Zarzour H, Kechida A, Taberkit AM. Vehicle detection from UAV imagery with deep learning: a review. IEEE Trans Neural Netw Learn Syst 2021; 3080276. https://doi.org/10.1109/TNNLS.2021.
[6] Duan H, Xin L, Chen S. Robust cooperative target detection for a vision-based UAVs autonomous aerial refueling platform via the contrast sensitivity mechanism of eagle's eye. IEEE Aerosp Electron Syst Mag 2019;34(3):18–30.
[7] Albanese A, Sciancalepore V, Costa-Perez X. SARDO: an automated search-and-rescue drone-based solution for victims localization. IEEE Trans Mob Comput 2021. https://doi.org/10.1109/TMC.2021.3051273.
[8] Ribeiro RG, Cota LP, Euzébio TAM, Ramírez JA, Guimarães FG. Unmanned-aerial-vehicle routing problem with mobile charging stations for assisting search and rescue missions in postdisaster scenarios. IEEE Trans Syst Man Cybern Syst 2021;3088776. https://doi.org/10.1109/TSMC.2021.
[9] Alladi T, Naren, Bansal G, Chamola V, Guizani M. SecAuthUAV: a novel authentication scheme for UAV-ground station and UAV-UAV communication. IEEE Trans Veh Technol 2020;69(12):15068–77.
[10] Wang H, Zhang Y, Zhang X, Li Z. Secrecy and covert communications against UAV surveillance via multi-hop networks. IEEE Trans Commun 2020;68(1): 389–401.
[11] Datta P, Sharma B. A survey on IoT architectures, protocols, security and smart city based applications. In: Proceedings of the 8th IEEE international conference on computing, communication and networking technologies (IEEE ICCCNT); 2017. p. 1–5.
[12] Bajaj K, Sharma B, Singh R, Rani S, Maheswar R, Kanagachidambaresan G, Jayarajan P. Integration of WSN with IoT applications: a vision, architecture, and future challenges. Integration of WSN and IoT for smart cities. Springer; 2020. p. 79–102. EAI/Springer innovations in communication and computing.
[13] Kim T, Qiao D. Energy-efficient data collection for IoT networks via cooperative multi-hop UAV networks. IEEE Trans Veh Technol 2020;69(11):13796–811.
[14] Sharma B, Obaidat MS, Singh K, Bajaj K. A comparative study on frameworks, MAC layer protocols and open research issues in internet of things. Ad Hoc Sens Wirel Netw 2019;45(3–4):275–91.
[15] Ye J, Zhang C, Lei H, Pan G, Ding Z. Secure UAV-to-UAV systems with spatially random UAVs. IEEE Wirel Commun Lett 2019;8(2):564–7.
[16] Lamport L, Shostak R, Pease M. The Byzantine generals problem. ACM Trans Program Lang Syst 1982;4(3):382–401.
[17] Barborak M, Malek M, Dahubra A. The consensus problem in fault-tolerant computing. ACM Comput Surv 1993;25(2):171–220.
[18] Alchieri EAP, Bessani A, Greve F, Fraga JS. Knowledge connectivity requirements for solving Byzantine consensus with unknown participants. IEEE Trans Dependable Secur Comput 2018;15(2):246–59.
[19] Cheng CF, Huang CW. The harmonized consensus protocol in distributed systems. J Supercomput 2019;75(11):7690–722.
[20] Cheng CF, Tsai KT. A flexible consensus protocol for distributed systems. IEEE Access 2019;7:90453–64.
[21] Kuo PC, Chung H, Chao TW, Cheng CM. Fair Byzantine agreements for blockchains. IEEE Access 2020;8:70746–61.
[22] Wang H, Fang H, Wang X. Safeguarding cluster heads in UAV swarm using edge intelligence: linear discriminant analysis-based cross-layer authentication. IEEE Open J Commun Soc 2021;2:1298–309.

[23] Fischer M, Paterson M, Lynch N. Impossibility of distributed consensus with one faulty process. J ACM 1985;32(4):374–82.
[24] Correia M, Bessani AN, Veríssimo P. On Byzantine generals with alternative plans. J Parallel Distrib Comput 2008;68:1291–6.
[25] Fischer M, Lynch N. A lower bound for the time to assure interactive consistency. Inf Process Lett 1982;14(4):183–6.

Chien-Fu Cheng is currently a professor with the Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung, Taiwan.

Gautam Srivastava is currently an associate professor with the Department of Mathematics and Computer Science, Brandon University, Brandon, Canada.

Jerry Chun-Wei Lin is currently a professor with the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway.

Ying-Chen Lin received the B.S. and M.S. degree in computer science and information engineering from Tamkang University, New Taipei, Taiwan.