EKR: An Efficient K-anycast Routing in UAV Networks

Kun Guo*, Liang Liu*, Wenbin Zhai* Youwei Ding†,
*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
Nanjing, China
Email: {gavin.k.guo, liangliu, wenbinzhai}@nuaa.edu.cn
†School of Artificial Intelligence and Information Technology, Nanjing University of Chinese Medicine
Nanjing, China
Email: ywding@njucm.edu.cn

***Abstract*—K-anycast refers to a communication model where a message is transmitted from a source to k destinations. Compared to multicast and unicast, k-anycast offers significant advantages in terms of load balancing, redundancy, and improved reliability. As far as we know, there is currently no research on k-anycast for UAV networks. We propose an Efficient K-anycast Routing scheme called EKR. The high dynamism of UAV networks poses a challenge in formalizing and addressing the k-anycast problem. Therefore, this paper utilizes predetermined trajectory information to construct Encounter Record Tree (ER-Tree), which converts dynamically changing network topologies into a static graph. Based on ER-Tree, we formalize the k-anycast problem in UAV networks and transform it into a Group Steiner Tree problem, which is known to be NP-hard. Then an efficient search algorithm is proposed to solve it and find the transmission path. To evaluate the performance of EKR, we performed simulations with the One simulator. The results demonstrate that EKR outperforms existing protocols in terms of delivery rate, end-to-end delay, and network overhead.**

***Index Terms*—UAV networks, multicast, anycast, k-anycast, trajectory-aware**

I. Introduction

With the development of monolithic integrated circuit (MIC) and imaging technology, unmanned aerial vehicles have greatly expanded their functions in civilian and military fields [1]. For the accomplishment of assigned tasks, UAVs need to corporately share information through the network and take orders from decision-making units. The key to support such operations is how to design a reliable and efficient routing algorithm.

The existing routing protocols for UAV networks can be roughly classified into unicast, broadcast, multicast, and anycast. Although extensive attention has been dedicated to unicast and multicast, there are few researches on anycast [2]. This paper focuses on a more generalized problem of anycast and proposes the k-anycast routing problem. In k-anycast, the source selects any k members from the k-anycast group with n nodes ($n \geq k$) to transmit data. Unlike conventional multicast, k-anycast doesn't require predetermined destinations. Interestingly, anycast and multicast are special cases of k-anycast, with k equal to 1 and n, respectively.

* Corresponding author
Email address: liangliu@nuaa.edu.cn (liang Liu)

K-anycast has numerous potential applications in UAV networks. In a large-scale network, a centralized station can become a bottleneck when all drones send their data to it, causing localized congestion [3]. To overcome this issue, k-anycast can be used to transmit the collected data to any k of the ground stations, thereby distributing the data traffic and avoiding congestion.

It is widely acknowledged that the replica mechanism plays a crucial role in ensuring fault tolerance and load balancing in network communication [4]. In the event of a network partition caused by high mobility or a single point of failure, it is imperative that the drone can be served by the remaining nodes. This is where k-anycast comes into play, selecting k highly available nodes for communication even in challenging situations. Once the nodes fail or exceed the service coverage range, k-anycast will swiftly and automatically reselect k available nodes, thereby enhancing the reliability.

Furthermore, in scenarios where a group of nodes with different resources and capabilities exists in a UAV network, k-anycast can be employed to select a specific number of nodes that, when combined, meet a threshold for providing equivalent functionality. By utilizing techniques like erasure coding [5], the UAV network can recover the original data even if some of the selected nodes fail or experience partial loss.

K-anycast offers several advantages over traditional multicast and unicast. Firstly, it provides more flexibility since the destinations can be dynamically selected based on needs, unlike multicast and unicast which typically require predefined destinations. Secondly, it results in higher transmission efficiency as the source node only needs to send data to any k nodes, helping to save communication cost and avoid network congestion. On the other hand, unicast communication involves sending separate copies of data to each recipient, which leads to a lower utilization of network bandwidth [6]. In conclusion, k-anycast is a promising technique with the potential to improve network performance and reliability in many scenarios. As far as we know, there is currently no research on k-anycast for UAV networks. Designing such an efficient k-anycast routing algorithm requires resolving the following challenges:

- How to efficiently sense the dynamic topology of UAV networks? The existing mechanisms rely on posteriori methods, which have to collect network information in real-time [7]. However, these methods consume a large amount of resources to maintain the topology and find routes, especially in dynamic UAV networks.
- How to find suitable routes for k undetermined destinations? K destinations selection and finding routes for them require complex algorithms that consider factors such as latency, congestion, and connectivity.

In this paper, we propose an efficient k-anycast (EKR) routing algorithm for UAV networks. Inspired by path planning algorithms [8, 9], we propose a priori-based and topology-aware strategy. This strategy utilizes the predetermined trajectories to build ER-Tree, which is simpler and more efficient than traditional posteriori methods, such as probe mechanisms and mobility prediction models. Additionally, we propose a search algorithm (EKRSearch) to find routes from the source to k destinations. The major contributions of this paper are summarized below:

- We propose an efficient data structure called ER-Tree, which converts dynamically changing network topologies into a static graph. It is easy to verify whether two nodes are reachable within the given time range by ER-Tree.
- We formalize the k-anycast problem and prove that it is NP-hard. To address this problem, the EKRSearch algorithm is proposed, which can efficiently select k destinations and find routes for them.
- To verify its performance, we perform simulations with the ONE simulator [10]. The results demonstrate that EKR outperforms existing protocols in terms of delivery rate, end-to-end delay, and network overhead.

The rest of the paper will consist of several parts. We present related work in Section II. Section III and IV provide the implementation details of EKR. Next, the evaluation of the k-anycast algorithm and a discussion of the results are presented in Section V. Finally, we conclude this paper.

II. RELATED WORKS

A. Unicast

Existing unicast routing protocols can be categorized as proactive protocols, reactive protocols, and hybrid protocols. Recently, some AI-enabled routing protocols were proposed for UAV networks. FLRLR [11] calculates the next hop route in real-time using fuzzy logic. This method reduces the average number of relays by using a future reward scheme and iterative learning. FESAIQ-Routing [12] uses simulated annealing (SA) optimization to control the rate at which it learns. As energy consumption becomes an important constraint in UAV communication [13], some efforts have been made to design energy-efficient routing protocols. ECaD [14] leverages movement information and residual energy level to enable the prediction of an imminent link failure, ensuring a robust level of communication stability. Some researches [15, 16] group UAVs in the same geographical area into multiple clusters. These cluster-based methods make the network more scalable, reduce energy consumption, and prolong the network's lifetime.

However, unicast is not suitable for multi-target transmission. Each additional stream will increase the bandwidth.

B. Multicast

The multicast protocols can be classified into tree-based ways and mesh-based ways. Recently, some scholars have suggested that the Global Positioning System (GPS) could be used to set up multicast paths. CLOM [17] assumes each node has a GPS device to identify the cluster to which it belongs. In an overlay multicast scheme, the cluster heads form a virtual mesh network, and the control information necessary for preserving the virtual topology is generated exclusively when modifications occur within the virtual topology. In SP-GMRF [18], one node should track the location of its neighbors, which will be used in the routing process. In a dynamic network, it is easier to use local information than to keep track of global routing information. Utilizing the geographical locations of nodes eliminates the additional overhead caused by control packets.

However, the GPS-based routing approaches are highly dependent on GPS functionality, such as the position accuracy of the feedback.

C. Anycast

AAODV [19] and ARDSR [20] are two anycast protocols based on traditional unicast protocols. They select targets from an anycast group based on the distance and route packets to the closest member. Lenders and May [21] devised a density-based anycast approach that considers both the proximity and the quantity of available anycast members during the routing decision-making process. To provide fast service with better Quality of Service (QoS), Budyal and Manvi [22] presented a method for multiple QoS constrained anycast routing, employing an Adaptive Neuro-Fuzzy Inference System (ANFIS). Likewise, MQAR [23] proposed a QoS-aware anycast routing method that takes mobility, congestion, and connection life cycle into account. k-anycast, as a more generalized model of anycast, is still in a relatively early stage. Some of those protocols can be found in [3, 7, 24].

In conclusion, there is no previous work studying the k-anycast problem in UAV networks, and conventional protocols cannot work well due to th high mobility of UAVs.

III. PROBLEM FORMULATION

The UAV network is represented as a weighted directed graph $G = (V, E)$, where $V = \{u_1, u_2, ..., u_i\}$ is a set of nodes, each representing a UAV, and $E = \{e_1, e_2, ..., e_i\}$ is a set of edges connecting pairs of nodes. An edge $e = (u_i, u_j, t_s, t_d)$ indicates that UAV u_i will meet UAV u_j at time t_s and t_d denotes the lifetime of the connection due to topology changes. The transmission rate is denoted as r (bytes per second) and the message size as b (bytes). Additionally, a

cost function C is defined over E such that each edge e_i in E has an associated transmission cost $C(e_i)$. The k-anycast routing problem can be defined as:

$$k_{any} = [s \rightarrow (D, k)]. \tag{1}$$

$s \in V$ is the source node of k_{any}. $D = \{u_1, u_2, ..., u_s\} \subseteq V$ represent all destinations in the k-anycast group. k is the specific threshold where $1 \leq k \leq |D|$.

We formally define the k-anycast routing problem as finding out the optimal transmission paths between the source and k destinations, with the goal of minimizing network transmission costs.

$$\text{minimize} \quad \sum_{i \in |E(P)|} C\left(e_i\right) \tag{2}$$

Subject to:

$$\sum_{d \in D} f\left(P, d\right) = k \tag{3}$$

$$\sum_{i \in |E(P)|} \pi\left(e_i, b/r\right) = |E(P)| \tag{4}$$

where

$$f\left(P, d\right) = \begin{cases} 1, & \text{if } d \in P \\ 0, & \text{otherwise} \end{cases}, \quad \pi\left(e, \eta\right) = \begin{cases} 1, & \text{if } e.t_d > \eta \\ 0, & \text{otherwise} \end{cases}$$

The function (2) states that the objective is to optimize the transmission costs. Here, P represents the generalized transmission paths and $E(P)$ represents the edges of P. (3) states that the paths generated should cover the exact k members of the given k-anycast group D. (4) states that each edge of the paths should exist until one message transmission has been completed.

Consider the special case where $k = |D|$, k-anycast can be reduced to a new problem: finding a minimum spanning tree that contains all specific nodes, which is known as the Steiner tree problem. Previous research [25] has shown that the Steiner tree problem is NP-hard. k-anycast, which generalizes the Steiner tree problem by involving the selection of k undetermined nodes to construct a constrained spanning tree, is also an NP-hard problem.

IV. THE PROPOSED K-ANYCAST ROUTING

In this section, we will introduce the details of EKR. Formalizing and addressing the k-anycast problem is challenging due to the high dynamism of UAV networks. Therefore, we leverage predetermined trajectory information to calculate the encounter windows when UAVs meet and depart from each other. During an encounter window, a communication link will be established and vanish upon separation.

To efficiently represent the encounter information, we propose a data structure called ER-Tree, which connects UAVs together based on the encounter windows within a certain period of time. Starting with a source UAV, each node in the ER-Tree represents a path from itself to the source. Finally, we propose an efficient algorithm (EKRSearch) for searching k-anycast transmission paths in the ER-Tree.

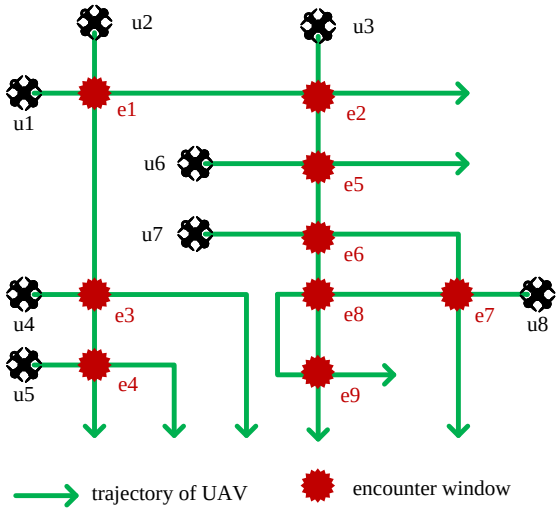


Fig. 1. An example of UAV trajectory.

As shown in Fig. 1, there are eight UAVs. They carried out flight operations based on the trajectory generated by path planning algorithms [8, 9]. At time 0, u_1 generates a message that remains valid for 40 seconds. A 2-anycast task is defined as sending the message from the source to any two destinations. Here, u_1 aims to send the message to any two destinations among u_4, u_5, and u_8. The delay constraint is set to the message's validity period (40s), which means that encounters occurring after 40 seconds will be ignored. We can calculate the encounter windows from u_1 during the time frame [0,40]s based on their pre-planned trajectories. The detailed information regarding these encounter windows is shown in Table I. Specifically, at window e_1, u_1 meets u_2 at 5s and the encounter lasts for 6s; at window e_2, u_1 meets u_3 at 20s and the encounter lasts for 5s; at window e_3, u_2 meets u_4 at 10s and the encounter lasts for 7s; at window e_4, u_2 meets u_5 at 14s and the encounter lasts for 5s.

TABLE I
ENCOUNTER WINDOWS

Window	UAV	UAV	Start time	Duration
e_1	u_1	u_2	5	6
e_2	u_1	u_3	20	5
e_3	u_2	u_4	10	7
e_4	u_2	u_5	14	5
e_5	u_3	u_6	22	7
e_6	u_3	u_7	26	6
e_7	u_7	u_8	28	4
e_8	u_8	u_3	45	5
e_9	u_8	u_3	55	6

Based on these encounter windows, EKR constructs an ER-Tree, which represents the spatiotemporal reachability between nodes. Spatiotemporal reachability refers to the ability to communicate between two nodes in a network within a certain time frame, taking into account both the physical distance and the time required for transmission. As depicted in Fig. 2, the

ER-Tree encompasses all feasible paths from the source to the destinations. According to the ER-Tree, there are three transmission paths that satisfy the 2-anycast condition:

1) $path_1$: $\{e_1, e_3, e_4\}$. The transmission path is $\{u_1 \rightarrow u_2, u_2 \rightarrow u_4, u_2 \rightarrow u_5\}$. u_1 can transmit data to u_2 during the time frame [5,11]s. u_2 can transmit data to u_4 during the time frame [10,17]s. u_2 can transmit data to u_5 during the time frame [14,19]s. The two destinations of 2-anycast are u_4 and u_5.
2) $path_2$: $\{e_1, e_3, e_2, e_6, e_7\}$. The transmission path is $\{u_1 \rightarrow u_2, u_2 \rightarrow u_4, u_1 \rightarrow u_3, u_3 \rightarrow u_7, u_7 \rightarrow u_8\}$. The two destinations of 2-anycast are u_4 and u_8.
3) $path_3$: $\{e_1, e_4, e_2, e_6, e_7\}$. The transmission path is $\{u_1 \rightarrow u_2, u_2 \rightarrow u_5, u_1 \rightarrow u_3, u_3 \rightarrow u_7, u_7 \rightarrow u_8\}$. The two destinations of 2-anycast are u_5 and u_8.

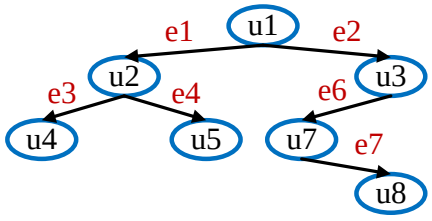


Fig. 2. An example of ER-Tree.

Among the three paths mentioned above, the transmission delays of $path_1$/$path_2$/$path_3$ are 14s/28s/28s. Among them, $path_1$ has the lowest delay. Furthermore, both receivers along $path_2$ and $path_3$ require individual copies from the sender, whereas $path_1$ reuses the $u_1 \rightarrow u_2$ route, utilizing fewer copies and conserving network bandwidth. Consequently, $path_1$ is selected as the transmission path for the 2-anycast routing.

The ER-Tree in the example has been pruned and trimmed to eliminate loops. However, as the number of nodes in the ER-Tree increases, listing all possible results and selecting the optimal solution becomes inefficient. Here after, the EKRSearch algorithm provides an efficient approach to finding k-anycast transmission paths. The construction process of the ER-Tree and the implementation of the EKRSearch will be described in detail later in this section.

A. Construction of the ER-Tree

The encounter record tree is generated by consolidating the scattered encounter windows and recording potential transmission paths from the source to the destinations. In this section, we will describe the process of constructing an ER-Tree based on these encounter windows.

- s represents the source UAV, which sends packages and the source of ER-Tree is $v_0 = (s, (s, s, 0, 0))$.
- Exp represents the transmission delay constraint. We set the value of Exp as the packet expiration time.
- Encounter window ew is denoted by (u_x, u_y, t_s, t_d), where u_x and u_y represent the two UAVs that encounter each other. t_s represents the start time of the encounter, and t_d represents the duration.

Algorithm 1 An ER-Tree Construction Algorithm

Input: Source UAV (s), encounter windows (EC), k-anycast group (Z), package size (b), transmission speed (r), delay constraint (Exp)
Output: An ER-Tree rooted at node v_0
Steps:

```
Initialize a priority queue q
Initialize root and set v_0 = (s, (s, s, 0, 0))
q.push(v_0)
while q is not empty do
    v_i ⟸ q.poll(), v_i = (u_i, ew_i)
    v_i.Candi ⟸ N(u_i, ew_i.t_s, Exp)
    for each encounter window ew_j ∈ v_i.Candi do
        if ew_j.t_d ≤ b/r then
            continue
        if ew_j ∈ EW(v_0) then
            continue
        u_j ⟸ otherUAV(ew_j, u_i)
        if u_j ∈ P(v_i) then
            continue
        create child node and set c(v_i) = (u_j, ew_j)
        q.push(c(v_i))
Trim(v_0, Z)
return v_0
```

- ER-Tree ERT is denoted by (V, E), where each node v is represented by a two-dimensional vector (u, ew). Here, u represents an UAV that can be reached by s directly or indirectly through multiple hops and ew is the encounter window between u and its previous hop. $ew.t_s$ must less than Exp to limit the height of the ER-Tree.
- During the growth process of ER-Tree, we need to search for candidate encounter windows until reaching leaf nodes. The candidate encounter windows are denoted as

$$v.Candi = N\left(v.u, v.ew.t_s, Exp\right) \tag{5}$$

where $N\left(v.u, v.ew.t_s, Exp\right)$ represent encounter windows that relate to $v.u$ and lie between $v.ew.t_s$ and Exp. As depicted in Fig. 2, the candidate encounter windows are $(u_2, u_4, 10, 17)$ and $(u_2, u_5, 14, 19)$ when considering u_2.
- $EW(ERT)$ indicates the encounter windows that have been used by current ER-Tree. Each encounter window can only be used once in the construction process.
- $P(v)$ represents the path from the source of ER-Tree to v, which is composed of multiple UAVs.
- $R(v)$ denotes the set of UAVs accessible from v. As depicted in Fig. 2, $R(v = (u_2, e_1)) = \{u_4, u_5\}$, which means that u_2 can reach both u_4 and u_5.

The details of construction is presented in Algorithm 1. The whole process involves expanding the tree by incorporating new nodes based on encounter windows, until there are no available candidate windows.

Algorithm 2 Trim

Input: ER-Tree root (v_0), k-anycast group Z
Output: null
Steps:

```
if v_0 is null then
    return
for each child node c of v_0 do
    if R(c) ∩ Z = ∅ then
        remove the subtree rooted at c from the ER-Tree
    else
        Trim(c,Z)
```

1) To begin, initialize the queue q and create the root node v_0, then insert v_0 into q.
2) If q is empty, the construction process ends. Otherwise, retrieve the first node at the head of the queue and designate it as the current node v_i to be processed.
3) Calculate the set of candidate encounter windows $v_i.Candi$, and use them to construct child nodes for v_i.
4) For each encounter window ew_j in $v_i.Candi$, if its duration $ew_j.t_d$ is less than transmission time, it will be discarded, and then go to Step 7. This constraint ensures two nodes have sufficient time to complete the data transmission task.
5) For each encounter window ew_j in $v_i.Candi$, if it has already been used in the construction process (i.e., $ew_j \in EW(ERT)$), discard it and go to Step 7 instead.
6) For each child node of v_i, check whether there is a cycle in the transmission path (i.e., $u_j \in P(v_i)$). If a cycle is detected, go to Step 7. Cycles not only introduce redundant branches to the ER-Tree, but also result in additional routing overhead.
7) For the eligible encounter window ew_j, create a child node and insert it at the end of q, then go to Step 7.

During the initial construction of the ER-Tree, numerous extraneous branches, which are not pertinent to k-anycast, are often generated and must be pruned. The pruning process is presented in Algorithm 2. For any node v_i of ER-Tree, if the subtree rooted at v_i does not contain any UAV of k-anycast group Z (i.e., $R(v_i) \cap Z = \emptyset$), it should be removed.

B. EKRSearch Algorithm

ER-Tree represents all possible paths from the source to the k-anycast group within a specified time range. The goal of k-anycast is to send data to any k destinations in the k-anycast group with the lowest possible transmission cost. In the context of ER-Tree, the objective is to find the minimum subtree that connects the source node to any k destinations.

As mentioned earlier, k-anycast is a generalization of the Steiner tree problem, which is known to be NP-hard. Specifically, when the scale of the network is large, the search space will exponentially increase. Therefore, based on previous research [26], we propose the EKRSearch algorithm.

EKRSearch is based on an iterative search for trees with low density, where lower density implies lower average transmission cost to reach the target. Each tree covers only a subset of destinations and the final solution is obtained by taking the union of them.

In the following part, we will introduce some key concepts of EKRSearch and describe the process of finding transmission paths in more detail.

- For a node v of ER-Tree, the weight of edge e between v and its parent can be expressed as $w(e)$:

$$w(e) = \alpha^{\frac{v.ew.t_s}{\max(t_s)} - \frac{v.ew.t_d}{\max(t_d)}} \tag{6}$$

  where $v.ew.t_s$ represents the start time of the encounter window, $v.ew.t_d$ represents the duration and α is a constant.
- Considering that the source UAV s can reach u_i through different paths, there may exist multiple nodes v in the ER-Tree associated with u_i (i.e., $v.u = u_i$). We group these nodes related to the same UAV as g_i. In this paper, u_i is one of the destinations in the k-anycast group. Given a k-anycast group $Z = \{u_1, u_2, .., u_n\}$, EKR creates a group for each destination and obtains a set $G = \{g_1, g_2, ..., g_n\}$.
- $T(v)$ indicates a subtree rooted at v. If $T(v)$ contains at least one node belonging to group g_i, it is said to cover g_i. If $T(v)$ covers at least one group of G, $T(v)$ is a covering tree for G, denoted as $T_v[G]$.
- The coverage number of $T_v[G]$ is denoted as $n(T(v))$. A k-covering tree is defined as $T_v^k[G]$.
- We define a metric called energy density (EDT) for evaluating routing paths. The energy density of a subtree $T(v)$ can be expressed as follows:

$$EDT(T(v)) = \frac{W(T(v))}{n(T(v))}. \tag{7}$$

  $W(T(v))$ represents the total weight of $T(v)$. Finally, the goal of EKRSearch is to find a k-covering tree from ER-Tree that covers exact k groups with minimal energy density.

The pseudocode of EKRSearch is shown in Algorithm 3. The input of the algorithm includes an encounter record tree rooted at v_0. The group set G to be covered and a given coverage threshold k. The algorithm returns the root of a k-covering tree $T_{v_0}^k[G]$. The k-covering tree represents the transmission paths used for k-anycast routing, which allows a source to send data to k destinations. The algorithm is expressed as follows:

1) Initialize T_{res}, k_{res}, $cover$, G_{res} and T.
2) The stopping condition for the recursion is that the input tree $T(v)$ contains only one node, which is a leaf. At this point, the single-node tree $T(v)$ is returned directly.
3) The process of identifying subtrees must be repeated until $k_{res} = 0$. The outer loop serves to indicate that it's not always possible for every search to locate subtrees that meet the k_{res} coverage requirement. This is due

Algorithm 3 EKRSearch

Input: $T(v)$,G',k'
Output: A k-covering tree $T_v^{k'}[G']$
Steps:

1: **Initialize:** $T_{res} = T(v)$, $k_{res} = k'$, $cover = \emptyset$, $G_{res} = G'$, $T = null$
2: **stopping condition If** v is a leaf **then** return $T(v)$
3: **while** $k_{res} > 0$ **do**
4: **recurse:** for every $c \in$child(v) and $k'' \in [1,k_{res}]$
$T_{c,k''}$ = EKRSearch($T(c)$,G_{res},k'')
5: **select:**(choose the result tree with the lowest density)
T_{aug} = MIN-DENSITY$\{T_{c,k''} \cup \{v\}\}$
6: **update**
(a)$cover$ = $cover \cup$ coverSet(T_{aug})
(b)k_{res} = k' - $cover.size$
(c)$T = T \cup T_{aug}$
(d)inactivate the nodes associated with T_{aug} from T_{res}
(e)remove the groups associated with T_{aug} from G_{res}
7: **end while**
8: **return** T

to either the fact that the number of g_i present in the search space is less than k_{res}, or that a T_{aug} with smaller coverage number, has a lower energy density.

4) Initiate recursive phases to construct subtrees. It traverses all child nodes of v, and for each child c, selects a coverage number, k'', from the range $[1, k_{res}]$, and makes a recursive call. In each recursive call, the incoming parameters are $(T(v), G_{res}, k'')$, and the computed subtree is referred to as $T_{c,k''}$.
5) Connect the various subtrees, $T_{c,k''}$, produced during the recursive phase, to v. The resulting tree, with v serving as the root, is referred to as an augmented tree T_{aug}. Subsequently, the algorithm selects the augmented tree with the minimum energy density.
6) Compute all groups covered by T_{aug} using coverSet(T_{aug}), record the result, and combine it with the current coverage set.
7) Update $k_{res} = k' - cover.size$.
8) Update $T = T \cup T_{aug}$.
9) Remove the nodes associated with T_{aug} from T_{res} and mark them inactive for the remainder of the loop process. Meanwhile, inactive nodes still contribute to the computation, albeit without considering their weight. The problem is reduced to finding the subtree with the minimum energy density among the remaining nodes, which must cover k_{res} groups.
10) Remove the groups associated with T_{aug} from G_{res}. Groups already present in the coverage set are to be ignored during processing.
11) Return T as the tree with the minimum energy density, and its coverage number matches the required k'.

It can be concluded by referring to the works of the literature [26]. The approximation rate of this algorithm can reach $O(h(T(v))\log k)$, while the time complexity reaches $O(\delta k^{O(h(T))})$. $h(T(v))$ represents the height of $T(v)$ and δ represents the maximum degree of $T(v)$.

C. An Example of EKR

In this section, we will utilize a comprehensive example to elucidate the intricacies of how EKR operates. The UAVs follow the trajectories generated by path planning algorithms, as depicted in Fig. 1. The encounter windows, obtained by computing the UAV's trajectory, are presented in Table I. The message originates from the source node u_1 and requires delivery to u_4, u_5, u_6, and u_7. With k set to 3, the message must be transmitted to any three of the four destinations, subject to a 60s delay constraint.

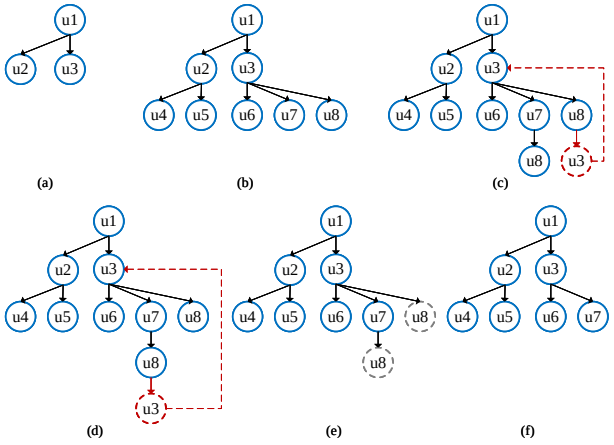


Fig. 3. The construction process of the ER-Tree.

We will construct the ER-Tree based on the encounter windows, and Fig. 3 depicts the whole process. As shown in Fig. 3(a), u_1 has two encounter windows $\{e_1, e_2\}$ that start within the interval $(0, 60)$s. Therefore, two sub-nodes u_2 and u_3 under u_1 are created and e_1 and e_2 are removed from the encounter window set. As shown in Fig. 3(b), u_2 has two encounter windows $\{e_3, e_4\}$ that start within the interval $(5, 60)$s. Therefore, two sub-nodes u_4 and u_5 under u_2 are created and e_3 and e_4 are removed from the encounter window set. u_3 has four encounter windows $\{e_5, e_6, e_8, e_9\}$ that start within the interval $(20, 60)$s. e_5 indicates that u_3 encountered u_6 at 22s, resulting in the creation of the sub-node u_6 for u_3. Similarly, e_6 indicates that u_3 encountered u_7 at 26s, leading to the creation of the sub-node u_7 for u_3. Since both e_8 and e_9 represent the encounter between u_3 and u_8, the earliest one in terms of start time (i.e., e_8) is selected, and the sub-node u_8 is created for u_3. Finally, e_5, e_6 and e_8 are removed from the set of encounter windows. As shown in Fig. 3(c), u_7 has one encounter windows $\{e_7\}$ that start within the interval $(26, 60)$s. Therefore, one sub-node u_8 under u_7 is created and e_7 is removed from the encounter window set. For the sub-node u_8 of u_3, there is one encounter window $\{e_9\}$ that is related to and starts within the interval $(45, 60)$s. e_9 represents

the encounter between u_3 and u_8 at 55s. However, it forms a loop between u_3, u_8, and u_3, so it is discarded. As depicted in Fig. 3(d), there is one encounter window $\{e_9\}$ for the sub-node u_8 of u_7, which is related to and starts within the interval $(28, 60)$s. However, it also forms a loop between u_3, u_7, u_8, and u_3, so it is discarded. As shown in Fig. 3(e), the tree is pruned for the 3-anycast zone of u_4, u_5, u_6, and u_7 by removing all u_8 branches.

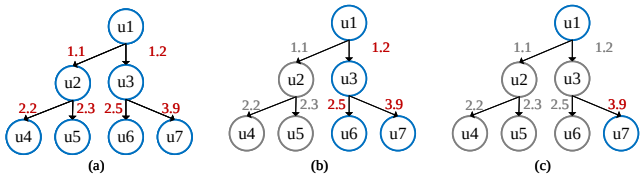


Fig. 4. The process of EKRSearch.

Based on the constructed ER-Tree, the EKRSearch algorithm searches the tree to find an efficient 3-anycast transmission path. Fig. 4 illustrates the search process of EKRSearch. To facilitate the description, we calculate the weight of edges in the ER-Tree using formula 6 in advance and display them in the figure. During a search, EKRSearch selects the subtree with the lowest energy density and returns it. Based on the ER-Tree generated using the aforementioned steps, calculate the energy density of 1-covering trees, 2-covering trees, and 3-covering trees:

1) 1-covering trees: $\{u_1, u_2, u_4\}$, the energy density is 3.3; $\{u_1, u_2, u_5\}$, the energy density is 3.4; $\{u_1, u_3, u_6\}$, the energy density is 3.7; $\{u_1, u_3, u_7\}$, the energy density is 5.1.
2) 2-covering trees: $\{u_1, u_2, u_4, u_5\}$, the energy density is 2.8; $\{u_1, u_3, u_6, u_7\}$, the energy density is 3.8.
3) 3-covering trees: $\emptyset$.

During the first round of selection, the 2-covering tree $\{u_1, u_2, u_4, u_5\}$ has the lowest energy density. Therefore, the 2-covering tree is removed from the ER-Tree, as shown in Fig. 4(b). Then, the coverage requirement k is updated to 1. In the second round of selection, the 1-covering tree $\{u_1, u_3, u_6\}$ has the lowest energy density. Therefore, the 1-covering tree is removed from the ER-Tree as shown in Fig. 4(c). Then, the coverage requirement k is updated to 0. When $k = 0$, the search process terminates, and the transmission path for 3-anycast is $\{u_1 \to u_2, u_2 \to u_4, u_2 \to u_5, u_1 \to u_3, u_3 \to u_6\}$.

V. PERFORMANCE EVALUATION

To analyze the performance of EKR, we implement the existing classical algorithms based on the environment of the ONE simulator [10].

A. Simulation Setup and Scenarios

In this paper, we present two experimental environments designed to evaluate the performance of EKR. In the first scenario, every UAV performs a periodic motion within a fixed area [27]. In the second scenario, all UAVs move randomly throughout the entire simulated environment.

k-anycast routing is a method of directing data from a source node to any k of n destinations. Throughout our experiments, we vary the value of k. Given that n is a variable across different simulation environments, we utilize a factor denoted as k/n. Table II provides a comprehensive overview of the default simulation settings.

TABLE II
DEFAULT EXPERIMENT PARAMETERS

Parameter	First Scenario	Second Scenario
Simulation Area	2000m × 2000m	2000 × 2000m
Simulation Time	480s	480s
Mobility Model	MapRouteMovement	MapRouteMovement
Communication Range	150m	150m
UAV Speed	8m/s	8m/s
Message TTL	200s	200s
Message Size	512bytes	512bytes
Message Creation Rate	4/s	4/s
Number of Destinations	4	8
Number of UAVs	17	30
Factor	0.75	0.75

B. Metrics and Compared Protocols

To evaluate the performance of EKR, we use the following four metrics.

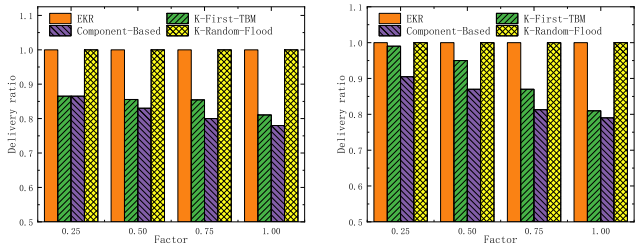
- Delivery rate: This metric measures the success rate at which messages travel from the source to the destinations. A higher delivery rate indicates that more messages are able to reach their destinations within the allotted TTL time.
- Average delay: This metric represents the average time taken for each message to be transmitted from the source to the destination.
- Overhead ratio: This metric refers to the overhead of the routing process, and it can be expressed as follows:

We compare EKR with three classical k-anycast routing schemes:

- Component-based k-anycast (CBK) [24]: CBK is a k-anycast routing algorithm designed for ad hoc mobile networks.
- TBM [28]: TBM is a multicast routing protocol based on UAVs trajectories. However, TBM does not originally support k-anycast. We have expanded TBM to K-First-TBM, which communicates with the first k nodes connected.
- Epidemic: Epidemic is a routing protocol designed to distribute messages to all neighboring nodes without requiring the maintenance of additional routing tables or multicast trees.

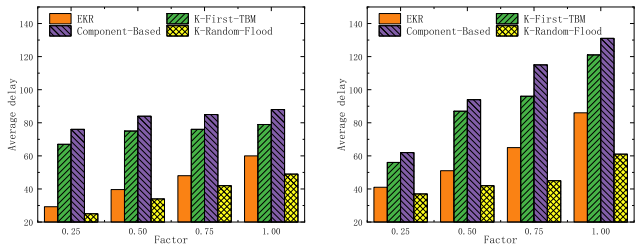
C. Analysis of Simulation Results

Impact of the Factor. As shown in Fig. 5, the delivery ratio of all three k-anycast tends to decrease as the factor increases since copies increase network congestion. As shown in Fig. 6, the average delay increases as the factor increases and this trend is particularly pronounced in the second scenario.



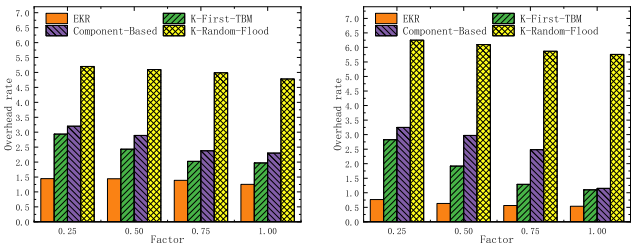
(a) the first scenario (b) the second scenario

Fig. 5. Impact of Factor on Delivery ratio.



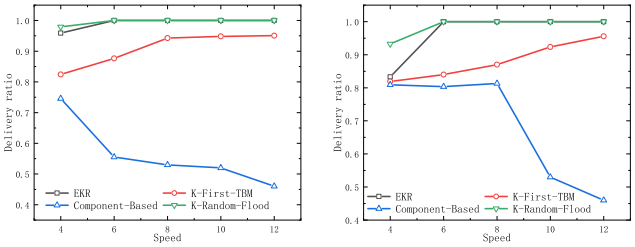
(a) the first scenario (b) the second scenario

Fig. 6. Impact of Factor on Average delay.



(a) the first scenario (b) the second scenario

Fig. 7. Impact of Factor on Overhead rate.



(a) the first scenario (b) the second scenario

Fig. 8. Impact of UAV Speed on Delivery ratio.

As shown in Fig. 7, the network overhead ratio of these implementations decreases slightly as the factor increases.

As shown in Fig. 5, K-Random-Flood and EKR achieve the maximal delivery ratio. Furthermore, K-Random-Flood and EKR show a smaller decrease, while K-First-TBM and CBK show a more pronounced decrease. EKR incorporates the trajectory information of unmanned aerial vehicles, ensuring the reliable forwarding of messages. However, CBK needs to maintain dynamic groups, which is unreliable.

As shown in Fig. 6, EKR and K-Random-Flood achieve a better average delay than CBK. Furthermore, when selecting k nodes, EKR takes into account both time delay and link reliability. K-Random-Flood, which takes into account the flooding characteristics, will try all possible paths. However, CBK lacks an optimal destinations selection mechanism and just consider hops.

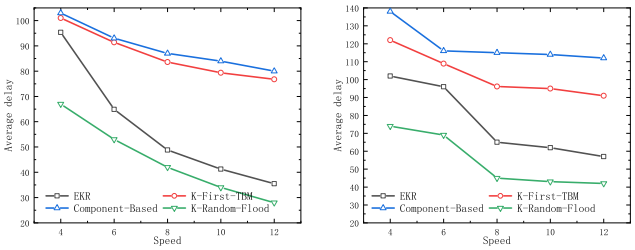
As shown in Fig. 7, the network overhead ratio of these implementations decreases slightly as the factor increases. EKR significantly outperforms the other protocols in terms of overhead ratio.

Impact of the UAV Speed. As illustrated in Fig. 8, the delivery rate of EKR gradually increases as the speed of the UAV increases, whereas the delivery rate of K-Random-Flood remains stable and changes less. This is attributed to EKR's utilization of trajectory information. The faster the node moves, the higher the likelihood of finding an end-to-end path. EKR can be optimized by exploiting the high mobility of UAVs. This is also where EKR is superior to traditional methods. The delivery rate of CBK decreases as the speed increases. This is because the high mobility of UAVs makes it difficult to maintain a group of k servers for CBK to forward the message to all destinations.

As shown in Fig. 9, it can be observed that as the speed of the UAV increases, the average delay decreases significantly. This is due to the fact that the faster the UAV node moves, the higher the connectivity between nodes, which leads to a quicker delivery of messages to their destinations. However, for CBK, the opposite is true as the speed increases, the average delay also increases. This is because high mobility makes it more difficult to maintain a group of k servers, resulting in a higher delay for message delivery.

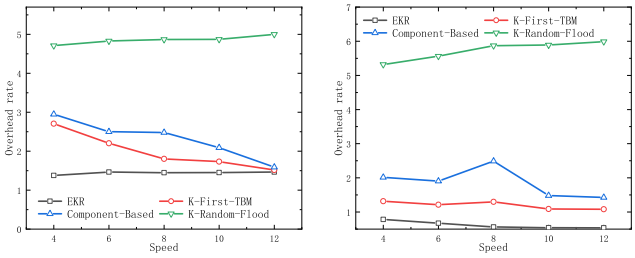
As depicted in Fig. 10, the network overhead ratio of K-Random-Flood, K-First-TBM, and CBK increases as the speed of the UAVs increases. However, for EKR, the overhead ratio decreases with an increase in speed.

Impact of the Number of UAVs. As demonstrated in Fig. 11(a), an increase in the number of nodes leads to a rise in the delivery rate for all four protocols. The higher density of nodes increases the likelihood of finding end-to-end paths from



(a) the first scenario (b) the second scenario

Fig. 9. Impact of UAV Speed on Average delay.



(a) the first scenario (b) the second scenario

Fig. 10. Impact of UAV Speed on Overhead rate.

source to destination. Notably, the impact on CBK is more significant than the others. Meanwhile, EKR and K-Random-Flood maintain a consistently high delivery rate with stability.

As depicted in Fig. 11(b), an increase in node density results in a decrease in average delay. This is because in high-density networks, constructing routes takes less time compared to sparse networks, thereby reducing delay. Notably, CBK has the highest average delay among the four protocols, while K-Random-Flood shows the lowest delay. EKR, on the other hand, exhibits comparable performance to K-Random-Flood.

Figure 11(c) illustrates that the overhead ratio of these protocols increases with the number of nodes. EKR exhibits the lowest ratio and shows no significant trend changes. In contrast, K-Random-Flood has the highest ratio and demonstrates a notable increase.

ACKNOWLEDGMENT

This work is supported by the Open Fund of Key Laboratory of Civil Aviation Smart Airport Theory and System, Civil Aviation University of China under No. SATS202206, the National Natural Science Foundation of China under No. U20B2050, Public Service Platform for Basic Software and Hardware Supply Chain Guarantee under No. TC210804A, the "National Key R&D Program of China" under No. 2021YFB2700500 and 2021YFB2700502, the National Natural Science Foundation of China under No. 82004499.

CONCLUSIONS

In this paper, we propose an efficient k-anycast routing for UAV networks, which routes data from the source to any k destinations within a group with n nodes. To sense the dynamic topology of the network, existing mechanisms mostly rely on posteriori methods such as proactive and reactive detection. However, they perform poorly and consume significant network resources in highly mobile UAV networks. Different from these methods, we propose a priori-based topology-aware strategy, which converts the dynamic network topology into an ER-Tree. Based on ER-Tree, we formalize the k-anycast problem into a Group Steiner Tree problem and propose an efficient algorithm to find the transmission path. We performed simulations with the One simulator. The results demonstrated that EKR outperforms existing protocols in terms of delivery rate, end-to-end delay, and network overhead.

REFERENCES

[1] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[2] R. Goścień, K. Walkowiak, and M. Klinkowski, "Tabu search algorithm for routing, modulation and spectrum allocation in elastic optical network with anycast and unicast traffic," *Computer Networks*, vol. 79, pp. 148–165, 2015.

[3] D. Gao, H. Lin, and X. Liu, "Routing protocol for k-anycast communication in rechargeable wireless sensor networks," *Computer Standards & Interfaces*, vol. 43, pp. 12–20, 2016.

[4] X. Wang, "Analysis and design of a k-anycast communication model in ipv6," *Computer Communications*, vol. 31, no. 10, pp. 2071–2077, 2008.

[5] Y. Hu, L. Cheng, Q. Yao, P. P. Lee, W. Wang, and W. Chen, "Exploiting combined locality for wide-stripe erasure coding in distributed storage." in *FAST*, 2021, pp. 233–248.

[6] S. Pathak and S. Jain, "A survey: on unicast routing protocols for mobile ad hoc network," *International Journal ofEmerging Technology and Advanced Engineering*, vol. 3, no. 1, pp. 204–210, 2013.

[7] X. Wang, J. Wang, K. Lu, and Y. Xu, "Gkar: a novel geographic k-anycast routing for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 5, pp. 916–925, 2012.

[8] Y.-H. Hsu and R.-H. Gau, "Reinforcement learning-based collision avoidance and optimal trajectory planning in uav communication networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 306–320, 2020.

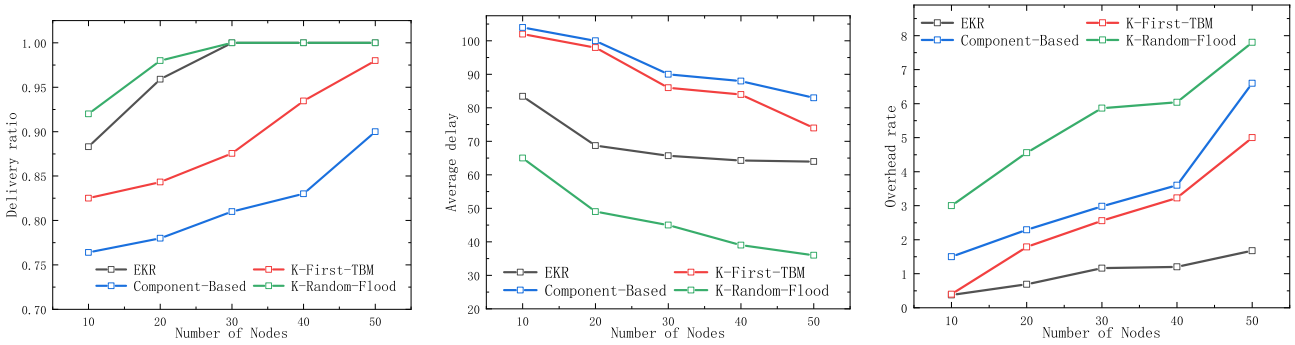
[9] P. Liu, H. He, T. Fu, H. Lu, A. Alelaiwi, and M. W. I. Wasi, "Task offloading optimization of cruising uav with fixed trajectory," *Computer Networks*, vol. 199, p. 108397, 2021.

[10] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*, 2009, pp. 1–10.

[11] C. He, S. Liu, and S. Han, "A fuzzy logic reinforcement learning-based routing algorithm for flying ad hoc networks," in *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2020, pp. 987–991.

[12] A. Rovira-Sugranes, F. Afghah, J. Qu, and A. Razi, "Fully-echoed q-routing with simulated annealing inference for flying adhoc networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2223–2234, 2021.

[13] A. Thibbotuwawa, P. Nielsen, B. Zbigniew, and G. Bocewicz, "Energy consumption in unmanned aerial vehicles: A review of energy consumption models and their relation to the uav routing," in *Information Systems*



(a) delivery ratio vs. number of nodes (b) average delay vs. number of nodes (c) overhead rate vs. number of nodes

Fig. 11. Impact of Number of Nodes.

Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology–ISAT 2018: Part II. Springer, 2019, pp. 173–184.

[14] O. S. Oubbati, M. Mozaffari, N. Chaib, P. Lorenz, M. Atiquzzaman, and A. Jamalipour, "Ecad: Energy-efficient routing in flying ad hoc networks," *International Journal of Communication Systems*, vol. 32, no. 18, p. e4156, 2019.

[15] S. Bharany, S. Sharma, S. Badotra, O. I. Khalaf, Y. Alotaibi, S. Alghamdi, and F. Alassery, "Energy-efficient clustering scheme for flying ad-hoc networks using an optimized leach protocol," *Energies*, vol. 14, no. 19, p. 6016, 2021.

[16] F. Aadil, A. Raza, M. F. Khan, M. Maqsood, I. Mehmood, and S. Rho, "Energy aware cluster-based routing in flying ad-hoc networks," *Sensors*, vol. 18, no. 5, p. 1413, 2018.

[17] H.-O. Lee, J.-S. Nam, and J.-H. Jeon, "Cluster and location based overlay multicast in mobile ad hoc and sensor networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 3, p. 687698, 2014.

[18] H. R. Hussen, S.-C. Choi, J.-H. Park, and J. Kim, "Predictive geographic multicast routing protocol in flying ad hoc networks," *International Journal of Distributed Sensor Networks*, vol. 15, no. 7, p. 1550147719843879, 2019.

[19] J. Wang, Y. Zheng, and W. Jia, "An aodv-based anycast protocol in mobile ad hoc network," in *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003.*, vol. 1. IEEE, 2003, pp. 221–225.

[20] G. Peng, J. Yang, and C. Gao, "Ardsr: an anycast routing protocol for mobile ad hoc network," in *Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication (IEEE Cat. No. 04EX710)*, vol. 2. IEEE, 2004, pp. 505–508.

[21] V. Lenders, M. May, and B. Plattner, "Density-based vs. proximity-based anycast routing for mobile networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. IEEE, 2006, pp. 1–13.

[22] V. Budyal and S. S. Manvi, "Anfis and agent based bandwidth and delay aware anycast routing in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 39, pp. 140–151, 2014.

[23] P. I. Basarkod and S. S. Manvi, "Mobility and qos aware anycast routing in mobile ad hoc networks," *Computers & Electrical Engineering*, vol. 48, pp. 86–99, 2015.

[24] B. Wu and J. Wu, "k-anycast routing schemes for mobile ad hoc networks," in *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2006, pp. 10–pp.

[25] I. Ljubić, "Solving steiner trees: Recent advances, challenges, and perspectives," *Networks*, vol. 77, no. 2, pp. 177–204, 2021.

[26] C. Chekuri, G. Even, and G. Kortsarz, "A greedy approximation algorithm for the group steiner problem," *Discrete Applied Mathematics*, vol. 154, no. 1, pp. 15–34, 2006.

[27] M. Asadpour, K. A. Hummel, D. Giustiniano, and S. Draskovic, "Route or carry: Motion-driven packet forwarding in micro aerial vehicle networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 843–856, 2016.

[28] J. Peng, H. Gao, L. Liu, N. Li, and X. Xu, "Tbm: An efficient trajectory-based multicast routing protocol for sparse uav networks," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International conference on smart city; IEEE 6th International conference on data science and systems (HPCC/SmartCity/DSS)*. IEEE, 2020, pp. 867–872.