electronics

MDPI

Article

Novel Framework-Based Routing for Task-Adaptive Mobile Networks of Unmanned Aerial Vehicular

Zhe Chu *, Zhijing Ye, Jiamiao Zhao, Linsheng He and Iftikhar Rasheed

Department of Electrical & Computer Engineering, University of Alabama, Tuscaloosa, AL 35487, USA; zye7@crimson.ua.edu (Z.Y.); jzhao15@crimson.ua.edu (J.Z.); lhe13@crimson.ua.edu (L.H.); irasheed@crimson.ua.edu (I.R.)

* Correspondence: zchu@crimson.ua.edu

Abstract: Many practical mobile ad hoc networks (MANET) have certain tasks instead of just randomly changing each node's positions. We call such a mission-driven network task-adaptive MANET. A typical example is the flying ad hoc network (FANET) that consist of unmanned aerial vehicles (UAVs), which may change its network topology based on different task requirements. Each node moves to new locations based on the targeted network shape. To maintain a smooth topology transformation and minimize the position changes, during shape change, a MANET typically keeps the core-area nodes more stable and allows the nodes in the outer area of the network to move more drastically. This means the entire network has an approximate framework that reflects the relatively stable nodes located in the core area. This research proposes a new routing scheme to quickly identify the optimal end-to-end path using the network framework extraction result. The proposed routing scheme ensures that the packets flow along the more stable network regions (thus with a lower packet loss rate). The framework extraction scheme is based on network shape geometry analysis for the median axis recognition. Our work has contributions to three aspects of realistic network protocol applications: (1) Provides a network multi-center election and member control methodology with detailed protocol design. (2) Creates a stable and reliable MANET framework extraction algorithm which aids in routing table generation. (3) Real-time Unix system protocol implementation and emulation based on Common Open Research Emulator (CORE) + Extendable Mobile Ad-hoc Network Emulator (EMANE). Simulation results indicate that our framework-based routing scheme outperforms a popularly used mobility-adaptive MANET routing scheme—OLSR (optimized link state routing)—in terms of throughput and delay.

Keywords: unmanned aerial vehicles (UAVs); routing; topology framework; network; EMANE; CORE; protocol design

Citation: Chu, Z.; Ye, Z.; Zhao, J.; He, L.; Rasheed, I. Novel Framework-Based Routing for Task-Adaptive Mobile Networks of Unmanned Aerial Vehicular. *Electronics* **2022**, *11*, 425. https://doi.org/10.3390/electronics11030425

Academic Editor: Christos J. Bouras

Received: 21 December 2021
Accepted: 27 January 2022
Published: 30 January 2022

1. Introduction

Task-adaptive unmanned aerial vehicular mobile networks (T-UAV Networks) contain devices (ex. UAVs) in the network with missions. We can also label this as mission-driven MANET with deployment for certain tasks, such as environmental surveillance, region coverage, sensing, etc. Each task requires certain network topology architecture. The topology's shape needs to transform to a new one as the task changes. As an example, a task-driven (mission driven) flying ad hoc network (FANET) that consists of unmanned aerial vehicles (UAVs) does not ask each node to fly randomly. Instead, each node flies alone in an approximate trajectory to maintain the entire network topology in a specific shape. Similarly, in a city surveillance application, the nodes may form a special shape to cover a city region. Such task-driven node mobility also helps to avoid unnecessary body collisions and task failure. Likewise, if we deploy some robot soldiers on the battlefield, these robots need to form certain desired troop shapes to meet the commander's requirements. Researchers also have enthusiasm to develop well-scheduled mission-driven robotic swarm

control algorithms. Ant-sized self-assembly robotics form different shapes as the robots operate different missions. During the whole process, some robots are more stable and serve as communication hubs. Others walk around for navigation and finish target jobs [1].

As a matter of fact, in practical *UAV* network applications, purely non-task-driven wireless networks are not popularly used. Based on different task requirements, nodes are often deployed with a certain initial spatial formation and the whole topology may change from one formation to another [2]. However, a task-driven network does NOT mean that each node has a pre-determined (fixed) trajectory, nor does it mean that all nodes' locations are pre-known and fixed. This is because most missions only have high-level task requirements, such as covering a city well, finding people who need rescue, etc. They do not necessarily require the setup of exact positions for each node. Each node can still move freely in its proximity as long as the entire network can maintain the approximate topology/coverage. Due to the inaccuracy of GPS in many weather conditions, it is difficult to specify the exact 3D positions for each ground or in-air node. This is also the reason that nodes need to use mutual coordination and communications to maintain the approximate network shapes and avoid physical collisions.

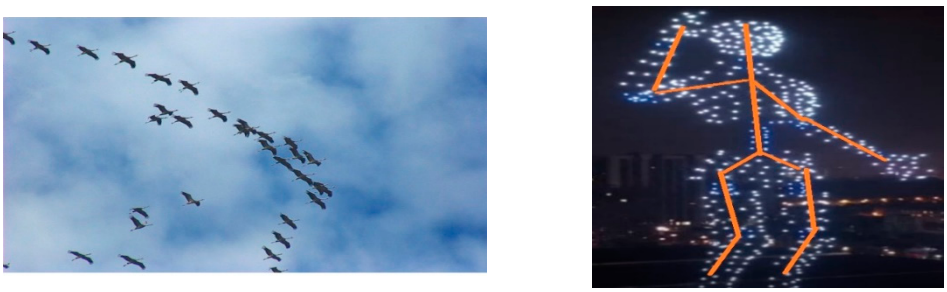
Therefore, even task-driven network nodes could have a certain amount of unpredictable mobility. For example, some flying nodes may have low batteries and eventually stop functioning. Then, other nodes need to coordinate to fill out those vacated small holes'. Some aircrafts may crash due to body collisions, and other nodes again need to compensate for those 'holes'. Aircrafts may change their relative positions for convenience in target hunting. Aircrafts can have frequent body shaking/tilt; they cannot stay in the same place for a long time.

We can summarize the main features of task-adaptive unmanned aerial vehicular mobile networks as follows:

(1) The nodes need to follow certain task requirements and form certain topology shapes/formations.
(2) The nodes do not move entirely randomly or aimlessly due to coordination requirements.
(3) Each node does not have a fixed, pre-determined position/trajectory.

To better understand the features of task-adaptive unmanned aerial vehicular mobile networks, here we use two analogies from other domains:

- Birds' migration (Figure 1a): When birds migrate to the south in winter, they maintain an overall flock shape. However, this does not mean that each bird must stay in a fixed location. Each bird can have certain position changes inside the flock. The T-*UAV* network's swarming concept borrows many features from the bird migration phenomenon by using certain attraction/repel force models [3].
- Drone animation (Figure 1b): In an amazing drone light show, although each drone can move freely in its local region inside the swarm, they maintain the overall shape of a human body [4].



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

Figure 1. Two task-driven systems with certain network shapes: (**a**) birds' migration; (**b**) drone formation.

In the real human body, it is the skeleton that acts as the body framework. Those skeleton bones act as the most stable parts of the whole human body. This is also our

motivation for using the concept of framework-based routing for T-UAV Networks. The framework concept facilitates topology maintenance, morphing, and data forwarding. The network framework is especially useful for mission-driven drone formations. These MANETs have specific geographical patterns. In most cases, the outer area of the entire network needs to move with a larger scope, compared to the inner area. In other words, the inner nodes have more stable positions. The 'skeleton bones' consist of those relatively more stable nodes and help to form the stable routing paths. By following the network framework, the established end-to-end routes have higher stability and outperform other proactive routing protocols, such as open link-state routing (OLSR).

In the proposed framework-based routing (FWR) scheme, some nodes are designated as master nodes that are evenly distributed in the entire network. As an example, in cluster-based network topology, the cluster heads (CHs) may serve as the master nodes. These master nodes calculate framework information and propagate the results to the nearby nodes. Eventually, all nodes get to know the network framework information. The framework consists of some 'bone' nodes that are aligned with each other to form the backbone ('trunk') of the framework to maintain backbone communications. Other non-framework nodes (called 'leaves') search the nearest born nodes.

As an example of the FWR concept, Figure 2 shows the identified framework for an example network topology. This proposed protocol can detect two categories of skeletons: trunks and stems, where trunks form the main routing paths of the network (used to transmit high-speed traffic), and stems help to connect 'leaf' nodes to the trunk(s). In the rest of the paper, the term 'framework' and 'skeleton' will be used in an exchangeable way.

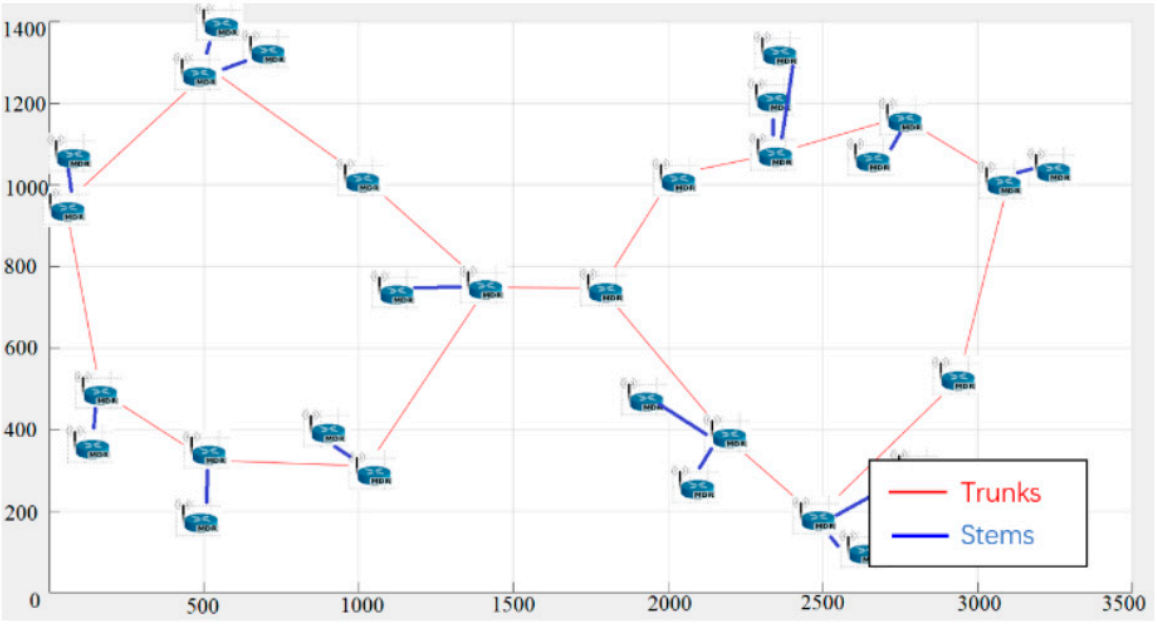


Figure 2. An example network topology with skeleton identified.

There are three stages in our FWR routing scheme: (1) gathering the position information, (2) calculating and distributing network skeleton information, (3) routing table update based on the detected framework. In the first two stages, two control messages are flooded in the network: position sharing packets (PSPs) and skeleton information packets (SIP). PSPs help to gather and distribute nodes' position information. They can also help to select swarm masters that are responsible for calculating and distributing the skeleton information. In stage III, every single node builds its routing table based on the skeleton nodes' positions.

The contributions of this paper are threefold:

(1) We propose the concept of FWR for T-UAV networks, since there exists a relatively stable framework in a task-adaptive network. A low-cost computer graphics algorithm is used to extract the skeleton of the network.
(2) Provide a MANET multi-center election and member control methodology with detailed protocol design. For a large MANET, we first select some nodes as masters and each master node performs skeleton identification for its cluster. Then, all masters exchange their results and form a complete global network framework.
(3) We have used a highly accurate network emulator, i.e., common Open Research Emulator (CORE) + Extendable Mobile Ad Hoc Network Emulator (EMANE), to implement and validate the quality of service (QoS) of the FWR protocol. Unlike many other network simulators (such as NS3, Riverbed, etc.), the protocol codes written in CORE+EMANE can be directly transferred to practical wireless network devices. This is mainly because CORE+EMANE provides a fully duplicate network hardware and software environment. CORE emulates independent Unix devices. The tested protocol will operate directly on these virtual Unix devices. EMANE provides a physical layer model that helps to decide whether a packet loss or bit error has happened when two Unix devices are communicating. We used CORE+EMANE to evaluate FWR's throughput and delay performance and compare it with a popularly used MANET routing scheme that is claimed to be able to tolerate high mobility—optimized link state routing (OLSR) [5]. Simulation results have shown that the proposed scheme has higher QoS performance.

The rest of this paper is organized as follows: Section 1 briefly summarizes some related works. The FWR stage I (position information gathering) via PSP messages will be discussed in Section 3. The master nodes selection scheme will be covered in Section 4. Next, in Section 5 we describe the skeleton detection algorithm (i.e., Stage II). Stage III (routing table building up) is explained in Section 6. Then, we provide the simulation results based on CORE+EMANE in Section 7, followed by the conclusions in Section 8.

2. Related Works

There are plenty of reasons that can cause the MANET routing scheme to select a traffic path with low link quality and even a failure. The reasons include but are not limited to: high dynamics, high wireless channel loss rate (bit error rate), and medium contention. Frequent low-quality paths or even path failure can trigger congestion in the network, which finally significantly decreases the QoS performance [6]. To detect and maintain stable traffic paths, there are three main types of MANET routing protocols that can be introduced: table-driven (proactive), on-demand (reactive), and hybrid schemes. All three categories of protocols have their pros and cons.

A reactive protocol finds a route after a communication task is given. Ad hoc on-demand distance vector (AODV) is one of the most popular reactive protocols. Since it reactively looks for a path only after a mission is given, it often involves an excessive amount of flooding messages and high latency. Many works have been conducted to improve such kinds of protocols. For instance, a scheme that combines the benefits of greedy peripheral stateless routing protocol (GPSR) and AODV, called PSO-GLFR, can shorten the path search delay [7]. It also limits the flooding behaviors and improves path selection.

Besides link-information-based reactive protocols, some researchers proposed location-information-based reactive protocols. Lin introduced a mobility-prediction-based geographic routing (MPGR) [8]. Although their results showed a great improvement when compared with AODV and GPSR, their position prediction requires some unrealistic assumptions.

To overcome the high latency and excessive flooding in reactive protocols, proactive protocols such as OLSR have been studied. In comparison with reactive protocols, proactive protocols pre-build the end-to-end path regardless of whether the communication tasks are given or not. Thus, they avoid on-demand path search delay. However, they require many maintenance messages for any active path. An improved OLSR is proposed by

enhancing the message flooding process and reducing control message size [9]. Some detailed performance analyses and enhancements for OLSR QoS and multipoint relay (MPR) nodes selection have been described in [10].

Besides improving previous well-known protocols, definitions of new matrices and new path searching methodologies are also often seen in research. To avoid link failure and maintain stable traffics paths, Zinal implemented a scheme named fault-tolerant routing [11]. Promiscuous listening is used for each single node to monitor neighbors' states and link signal noise ratio (SINR). Link quality metrics are defined by considering both physical interference (realistic signal interference) and logical interference (hidden terminal problem). Local repair and reroute schemes are built based on link quality metric value changes. The significant issue for Zinal's work is that they assumed that CSMA-CA-based protocol is running on the MAC sub-layer, which means their work is not extendable for the MANET with other lower layer designs. Similarly, Guo also introduced new matrices to help select paths in their works [12]. Six measurement criteria have been defined to describe the network topology situation at a specific time point: transmission distance metric, node connection reliability metric, progress metric, direction metric, network delay metric, and node lifetime metric. The path detection problem is finally converted into an optimization problem, from which with optimized paths can be calculated. Despite how well the metrics are designed, solving a mathematical optimization problem is extremely time consuming and requires devices with strong computation ability. Hence, Guo's work is more biased towards theoretical proof and far from realistic.

Our proposed routing scheme is a location-information-based proactive protocol and its design can use the extracted skeleton information to speed up the path establishment process. The skeleton nodes may have changes in different missions. However, the overall framework is always used for low-latency data routing. In addition, a detailed packets-level design is provided, which can be further extended to realistic protocol implementation.

3. Position Information Collection

Given that skeleton extraction requires approximate position information, gathering location information of each node is a necessary step. MANET nodes' relative positions can be calculated through receive signal strength (RSS) and a well-defined signal strength decay model, i.e., RSS $\propto 1/d^2$. In this section, we will introduce the structure of our defined message, i.e., position sharing packet (PSP), and explain the position information renewal policy. This belongs to stage I in our routing scheme.

3.1. Position Sharing Packets

In our FWR protocol implementation, PSP has a packet format as shown in Figure 3. It is used to help to gather location information of the network. A PSP contains a table with six columns. Each row in the table records location information for one node. The leftmost column represents the node's IP address. The following three columns are relative cartesian coordinates (X, Y, Z). $Z = 0$ means 2D network. The Time column helps to keep updating each record in the table. The last column, Vote Ticket, is used to select master nodes (details in Section 4).

IP	*X*	*Y*	*Z*	*Time*	*Vote Ticket*
10.0.0.3	563.0	385.0	0	1594306852.71	10.0.0.4
10.0.0.2	459.0	224.0	0	1594306854.10	10.0.0.4
10.0.0.4	680.0	149.0	0	1594306851.54	10.0.0.4
10.0.0.5	850.0	64.0	0	1594306852.13	10.0.0.4
10.0.0.6	665.0	521.0	0	1594306851.42	10.0.0.4

Figure 3. Position table in FWR protocol Stage I.

3.2. Local Information Updating Policies

In our protocol, each node opens a socket to keep listening to its neighbors' PSPs and updates its network information in its local buffer. Whenever a PSP arrives at a node, the node needs to compare the information saved in the PSP and its local buffer. The following three policies will be executed to keep updating the position information saved in the local buffer:

(1) Non-existing records will be added to the local buffer.
(2) If records with the same IP show up, only the record with a larger time value will be kept.
(3) The local buffer will be checked periodically, and outdated records will be deleted.
(4) Through the three above position information updating policies, each node obtains the most recent network position information. Policy (3) also helps each node produce a quick response whenever a node is removed.

4. Network Clustering and Master Selection

There are usually two types of structures for MANET: flat and hierarchical. A flat structure is a network with nodes that have equal status. The flat network usually has small size and limited expandability. In contrast, the hierarchical structure refers to assigning a cluster head node(s). Compared with the flat structure, hierarchical has flexible scalability and fewer control costs, which in turn has potential to provide better QoS performance.

Previous researchers have provided lots of hierarchical structure creation methodologies. The WCA weighted algorithm considers the speed, energy, connecting nodes, and the degree of restrictions to calculate an evaluation score for each node [13]. Nodes with higher scores with be selected as cluster heads. In the minimum ID algorithm, each node broadcasts its ID values to its neighbors' nodes periodically [14]. This approach focuses on security issues, which can significantly improve network's ability to defend against attacks. Passive multi-hop clustering algorithm (PMC) focuses on node clustering under high-speed scenarios, in case traditional clustering algorithms perform poorly in terms of stability and reliability [15].

We also propose the use of a distributed approach to create a hierarchical MANET structure. However, our clustering method focuses more on realistic protocol implementation. Firstly, piggybacking master node election information on flooded packets helps to pick center control master node(s). In the next step, those cluster heads (CHs) perform local skeleton extraction for their local area. Then, all masters exchange their results to produce a larger scale of skeleton information.

4.1. Single Master Selection

To select a suitable master from a network region, we first identify the geographic center of the region. Then, we find the node that is closest to the geographic center and select it as the master node. Figure 4 shows an example of finding a single master *UAV* for a swarm.

The position table illustrated in Figure 3 is used for the master node selection. The last column of the table, named 'Vote Ticket', provides information for master *UAV* selection. Before a node distributes its local position table to its neighbors, it calculates the geographic center for the swarm and computes the swarm master. Then, the local node adds its selected master UAV's IP address to the 'Vote Ticket' column. In Figure 3, we can see that all nodes selected the node with IP address 10.0.0.4 as the master node. A program thread has been built to periodically check the local position table. If the local node has the highest percentage of votes, it is selected as the master node.

4.2. Multiple Masters Detection

To select multiple masters, a large network can be separated into sub-groups (clusters). Each cluster then selects its own master. Figure 5a,b shows two examples of swarm

clustering and master selection. Figure 5a has two clusters; hence, two masters were selected. Similarly, Figure 5b has three clusters with three masters.

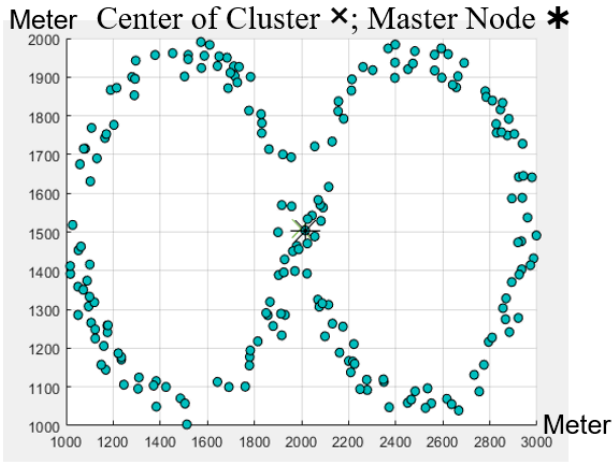
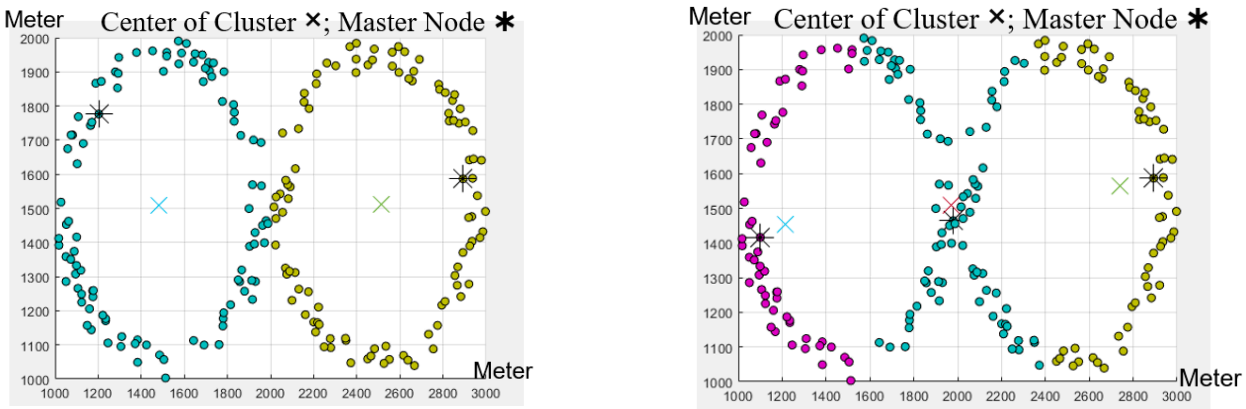


Figure 4. Single master election.



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

Figure 5. Multi-master cases: (**a**) Two-masters case. (**b**) Three-masters case.

Classification is a grand topic belonging to computer science and hot AI topics. For unsupervised learning (classification without labels), there are 4 most often seen classification algorithms: DBSCAN, OPTIC algorithm, K-mean, and hierarchical. The first two DBSCAN and OPTICS algorithms are both density-based algorithms; the remaining two are distance-based algorithms which is perfect fit in our case. The hierarchical clustering sequentially merges similar clusters. It means the algorithm is greedy and reaches the sub-optimum level in each step, which results in a sub-optimal solution. In contrast, K-means tries to reach global optimum or easier for network protocol implementation. Simple flooding method can reach the goal of implementing K-means clustering. In this paper, we simply used the K-means clustering algorithm to identify those clusters. In addition, adding mission ID to each device can help to generate better classification results. However, it belongs to the hot topic of supervised learning, which involves hot AI topics. We left it for future work and tried to stay on network protocols for this paper.

To implement the multi-master selection process, the Vote Ticket column in the position table (Figure 3) is preferred. Each node adds the calculated master's IP address to the Vote Ticket column. Periodic checking of the local position table is needed. If there are K clusters in the swarm, the top K candidate masters are chosen as the masters. Moreover, clustering information is included in the skeleton information packets (SIP) and distributed into the network. In Section 5, we will provide more descriptions of the use of SIP.

5. Network Skeleton Detection

Through the analysis of the position information in the position table by using a computer vision skeleton detection algorithm, the masters can extract the rough network framework. Fine enhancement for the skeleton is also needed to make sure that the network skeleton satisfies communication network requirements, such as delay constraint. Stage II of the protocol is responsible for the skeleton detection and skeleton information distribution. An enhanced network skeleton detection algorithm is built to take the position information (shown in Figure 2) as the inputs, and output two skeleton information matrices, as shown in Figure 6. These two matrices are named as skeleton node index matrix (SNIM) and skeleton node connection representation (SCRM), respectively.

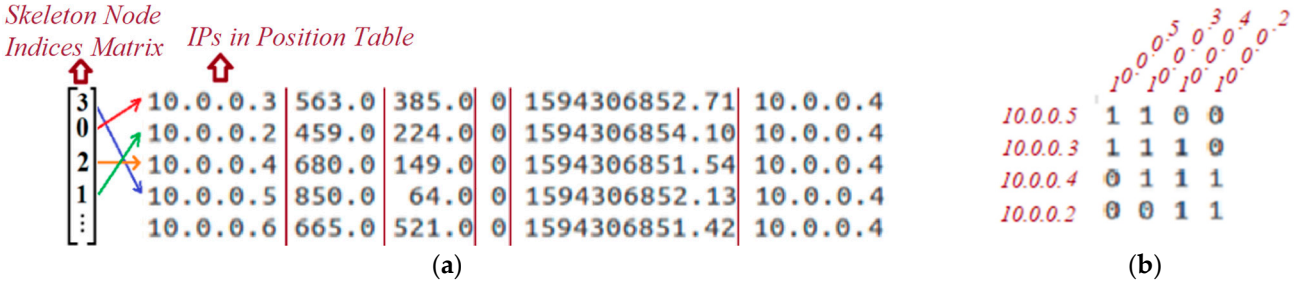


Figure 6. Skeleton information matrices: (**a**) skeleton node indices matrix (SNIM); (**b**) skeleton connection representation matrix (SCRM).

5.1. SNIM and SCRM

The skeleton node index matrix (SNIM) is a one-column matrix containing indices that point to different IPs (see Figure 3, the position table). In Figure 6a, the first number '3' means that the fourth IP address 10.0.0.5 in the position table is one of the skeleton nodes.

The skeleton node connection representation matrix (SCRM) contains two possible values: 0 and 1. Value 1 indicates a direct link between two skeleton nodes. Figure 6b, below, shows an example of an SCRM. Each (row, column) tells whether a skeleton node has a direct link with other skeleton nodes. For example, the first row indicates how 10.0.0.5 connects with other skeleton nodes. The first '1' in the first row means that 10.0.0.5 can talk to itself. The second '1' in the first row tells that 10.0.0.5 has a direct connection or link with the skeleton node 10.0.0.3.

5.2. Computer-Graphics-Based Point Cloud Skeleton Detection Algorithm

The network shape contour detection methods mainly use computer version algorithms. There are some challenges in shape detection: (1) Incomplete set: Due to time and memory limits, the collected node information may be incomplete, which means that the median skeleton must be detected even with incomplete surface places (Figure 7). (2) Noisy values: Due to the GPS accuracy limit, each identified location is often a noisy value, which could be described by a Gaussian distribution with the mean as the actual location. The noise variance could be large in some locations. Then, how do we still detect the actual skeleton axles under such noisy localization results? (3) Erroneous values: A small number of internal nodes (i.e., not belonging to the boundary of the whole network shape) may be mistakenly regarded as the boundary nodes in a network shape. Thus, the topology has 'outlier' nodes. A good skeleton extraction algorithm should be able to overcome the effects of those outlier points. (4) Distributed, incremental computations: When we convert the computer version algorithms to meaningful network protocols, all the computations must be done in a distributed way (here we use masters to calculate each local region's skeleton), i.e., each node can only use localized computations without global control.

We are the first team to design a skeleton extraction protocol to overcome all of the above four issues. In particular, our proposed scheme can correctly recognize the skeleton axle positions under noisy, incomplete, and outlier topology (see Figure 7).

In the computer graphics field, the node system with Cartesian coordinates is known as the points cloud. There are many points-cloud-based skeleton detection algorithms. Huang introduced a robust, distributed, and noise-tolerant algorithm called L1-medial skeleton, which is used as our point cloud rough skeleton detection algorithm [16]. In general, the point cloud skeleton detection problem can be generalized as solving the math system below:

$$a\ set\ of\ unoriented\ points\ Q = \{q_j\}\ and\ j \in J \tag{1}$$

$$a\ set\ of\ unoriented\ points\ X = \{x_i\}\ and\ i \in I \tag{2}$$

$$find\ argmin\Sigma_{i\in I}\ \Sigma_{j\in J}\|x_i - q_j\| \tag{3}$$

where Q is the set of point cloud positions, which can be represented as cartesian coordinates for each node. X is the skeleton location information set, which should also be represented in cartesian coordinate format.

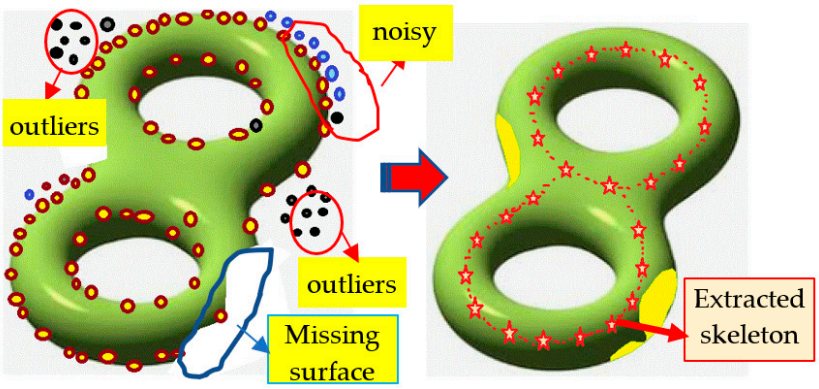


Figure 7. Skeleton extraction scheme.

For the math system represented by the above equations, the problem can be regarded as finding the location of x_i to minimize the total Euclidean distance to a set of points, Q. In [16], Huang proved that a better solution can be reached by replacing the minimization function (Equation (3)) with the following one:

$$argmin\Sigma_{i\in I}\ \Sigma_{j\in J}\|x_i - q_j\| * \theta(\|x_i - q_j\|) + R(x) \tag{4}$$

The weight function θ and regulation term R can be expressed as below:

$$\theta(y) = \exp(\frac{-y^2}{(h/2)^2}) \tag{5}$$

$$R(x) = \Sigma_{i\in I}\ \gamma_i \Sigma_{i'\in I\setminus\{i\}} \theta(\|x_i - x_{i'}\|) * (\|x_i - x_{i'}\|^0 - \|x_i - x_{i'}\|^1 - \|x_i - x_{i'}\|^2)/(\|x_i - x_{i'}\|^2) \tag{6}$$

Here, γ_i and h are the balancing constant and support radius, respectively. By solving the adjusted problem, the point set X can be expressed as an iteration format, with each single element x_i expressed as a function of previous iteration results. From the perspective of network skeleton detection, the algorithm complexity of the practical implementations needs to be considered. It is challenging to convert a computer vision algorithm to a practical distributed network protocol. For example, we need to use the minimum number of joint nodes to construct a connected network shape for efficient skeleton extraction, while making sure that all 'leaf' nodes are covered within the joint nodes' radio frequency (*RF*) ranges.

5.3. Algorithm Enhancement

There are three steps in our protocol that uses points cloud for network skeleton detection: (1) data normalization, (2) random sampling, and (3) converting the computer

vision algorithm to practical protocol. We explain below the methodology to convert a computer vision algorithm to the skeleton detection scheme.

The first step of skeleton detection is data normalization, which aims to remove the data redundancy. It makes the algorithm work for any point cloud size by shrinking or expanding all data values to the standard range of −1 to 1 (or from 0 to 1). Therefore, the data normalization makes the algorithm independent of the actual size of the system.

Figure 8 shows an example swarm pattern. The pattern has a size of 100 m × 50 m. We enlarged the same pattern by factors of 10 and 100, respectively, to get two more example swarms with sizes 1000 m × 500 m and 10,000 m × 5000 m, respectively. Their normalized data results are shown in Figure 8. Notice that, even with different network sizes, the same position pattern will have a similar normalized result.

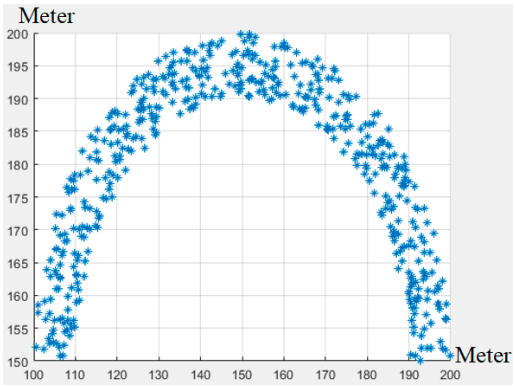
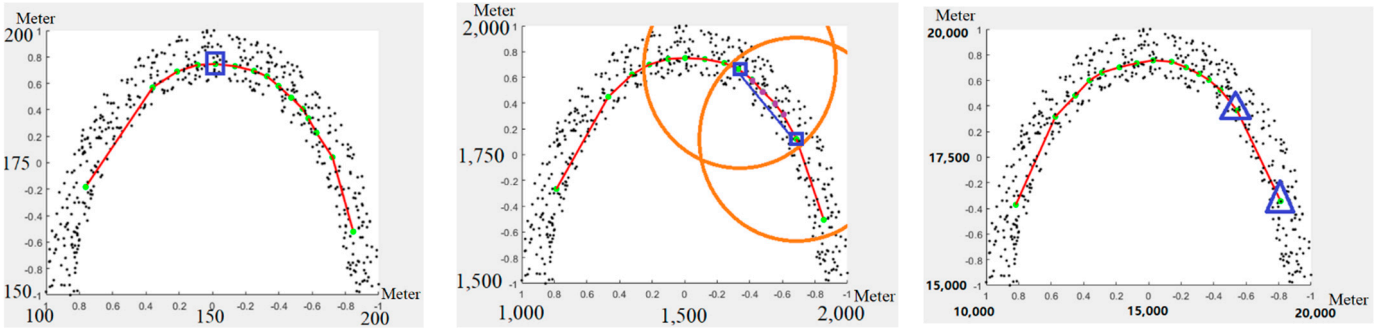


Figure 8. Swarm patterns with size 100 m × 50 m.

Figure 9a shows the results of applying the skeleton detection algorithm to the topologies shown in Figure 8. Figure 9b shows the result with swarm size 1000 m × 500 m, and Figure 9c shows the result with swarm size 10,000 m × 5000 m. The skeleton nodes are highlighted with large dots, and the links among them are highlighted with lines. As we can see, with the same normalized input data, the skeleton detection algorithm generates similar results with different sizes of the points clouds. We can call such results rough skeletons.



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

Figure 9. (**a**). Extracted skeleton for network size 100 m × 50 m; (**b**) size 1000 m × 500 m; (**c**) 10,000 m × 5000 m.

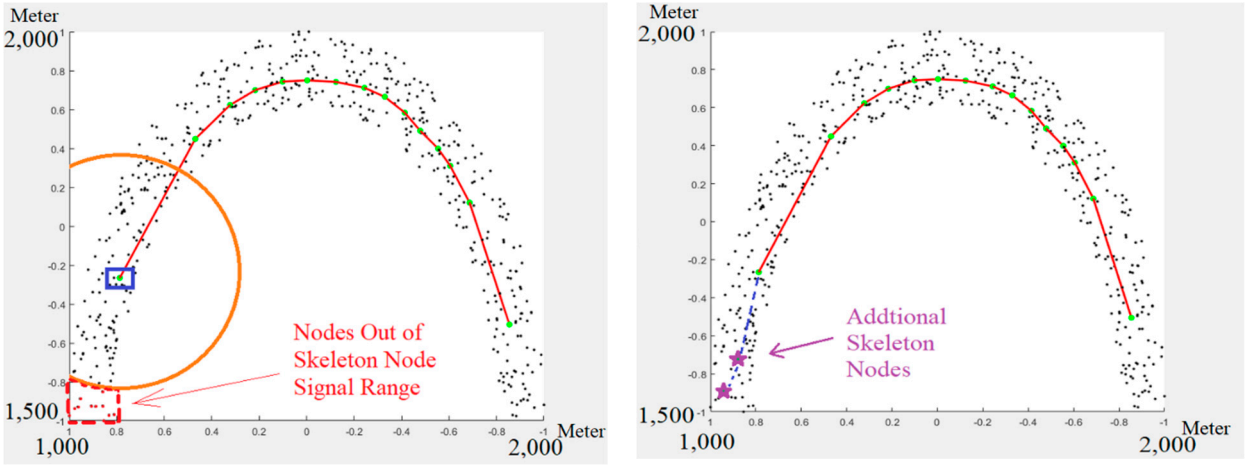
The skeleton detection algorithm may generate redundant skeleton nodes. The circles in Figure 9b show the *RF* signal ranges for the nodes marked as small blue rectangles. The skeleton nodes between them are redundant from a skeleton routing perspective, since the two blue nodes already have a direct *RF* link between them.

In Figure 9a, the node marked with the rectangle box can directly reach all other nodes in the entire network. Hence, we expect that the detected skeleton converges to a single

node. Redundant skeleton nodes can add more hops to the end-to-end transmissions. This decreases the network throughput. Therefore, unnecessary skeleton nodes should be removed.

On the other hand, a skeleton detection algorithm that ignores the actual size of the network may generate a result that lacks necessary skeleton nodes. In Figure 9c, the skeleton nodes marked with triangles have a distance larger than the signal range (600 m). Some intermediate nodes need to be added between them to enable all skeleton nodes to reach each other.

Another case that lacks skeleton nodes is shown in Figure 10a. Here, the skeleton node in the left bottom corner is highlighted by the small rectangle box. Since the signal range is 500 m, the nodes in the left bottom area (highlighted with dash lines) are not able to build a direct link with any skeleton node. To establish a direct link between the skeleton node and a nearby leaf node, more skeleton nodes need to be added here. The nodes marked with stars, shown in Figure 10b, have been added to solve the above issue. Here, some nodes are selected from non-skeleton nodes and promoted as skeleton nodes.



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

Figure 10. (**a**). Nodes out of skeleton nodes' *RF* range; (**b**). added skeleton nodes.

To reflect the above issue, we re-constructed the skeleton extraction math model as follows:

$$UAV\ Position:\ a\ set\ of\ unoriented\ points\ Q = \{q_j\}\ and\ j \in J \tag{7}$$

$$Network\ Skeleton\ Position:\ a\ set\ of\ unoriented\ points\ X = \{x_i\}\ and\ i \in I \tag{8}$$

To find extreme point X using the following optimization problem:

$$Objective\ Function:\ min\ \Sigma_{i\in I}\ \Sigma_{j\in J}\|x_i - q_j\| * \theta(\|x_i - q_j\|) + R(x) \tag{9}$$

$Constraints:$

$$\|x_i - x_{i-1}\| \leq RF\_range\ for\ \forall\ i \in I \tag{10}$$

$$max\|x_i - q_j\| \leq RF\_range\ for\ \forall\ i \in I, j \in J \tag{11}$$

$$Y = \{x_1, x_2, \ldots, x_{l-1}, x_{l+1}, \ldots x_i\}\ is\ not\ a\ feasible\ soulution \tag{12}$$

Equations (7) and (8) indicate skeleton nodes and *UAV* position are all 3D cartesian coordinates. The above math model tries to find a feasible solution that optimizes the objective function. Equation (9) is the objective function, which is the same as the L1-medial skeleton. This function can be replaced by another computer vision skeleton detection algorithm. Constraint (10) claims that any connected skeleton node should not have an

interval distance larger than the *RF* range. Constraint (11) suggests the maximum distance from any general node to its closest skeleton node should be less than the *RF* range. The last constraint (12) indicates that removing any skeleton node from the current skeleton node set X will make the system infeasible. Hence, X contains the minimum skeleton node set to cover all general nodes within the *RF* range. Redundant nodes should be removed from the solution set X.

Instead of solving the above math model and finding the exact skeleton set, a more feasible solution (i.e., sub-optimal solution) that satisfies all the constraints can save much computation time. This is critical from the perspective of network protocol overhead. The main pseudocodes of the skeleton detection algorithm are shown in Algorithm 1. In Lines 1~4, the computer-vision-based skeleton detection algorithm is executed based on the swarm position information. The skeleton node indices matrix (SNIM) is used in the algorithm. In Line 5, the skeleton nodes connect based on the *RF* range limit. Line 6 checks whether the situation of Figure 9c has occurred. If yes, more skeleton nodes will be added. Line 8 checks if the situation of Figure 9a,b has occurred; if so, the redundant nodes will be removed. Line 10 checks whether the situation of Figure 10a has occurred. More skeleton nodes will be added for this situation. Line 14 produces the final outputs, which are two matrices with network skeleton information.

Algorithm 1. Network skeleton detection algorithm (pseudocodes).

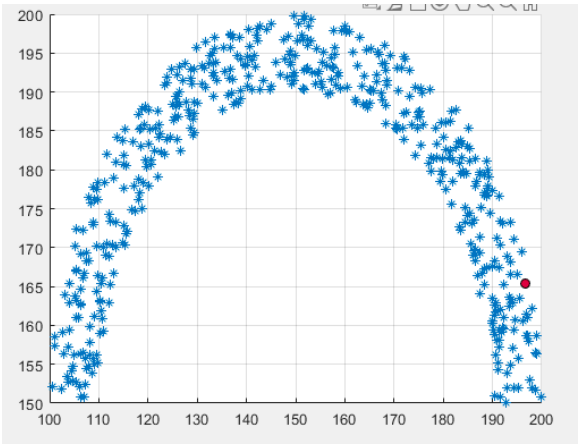
1	Program starts
2	Data normalization
3	Random sampling
4	Apply computer-vision-based skeleton detection algorithm. Output skeleton node indices matrix SNIM.
5	Calculate how skeleton nodes connect with each other based on signal range. Output skeleton node connection representation matrix SCRM.
6	If SCRM shows a termination in the middle of the skeleton node:
7	(Lack of skeleton nodes.) Add skeleton node in break-out areas. Renew SNIM and SCRM.
8	If any skeleton record in SCRM is a subset of another skeleton node:
9	(Redundant skeleton node detected.) Delete unnecessary nodes' records from SNIM and SCRM.
10	If there exists a node that cannot build a direct link with other skeleton nodes:
11	(Nodes cannot be reached by skeleton nodes already detected.) Random pick up a node from broken areas as skeleton node.
12	Add new skeleton node into SNIM.
13	Go to line 5.
14	Output network skeleton information saved in SNIM and SCRM.

Figure 11 shows our algorithm results for the swarm with sizes from small to large. As we expected, the smallest swarm has a single skeleton node (located on the right bottom corner) since the pattern converges into a single dot. The largest swarm has a complex structure, with all nodes covered within the skeleton nodes' *RF* signal range. Figure 12 shows the skeleton for a large network (10,000 m $\times$ 5000 m).

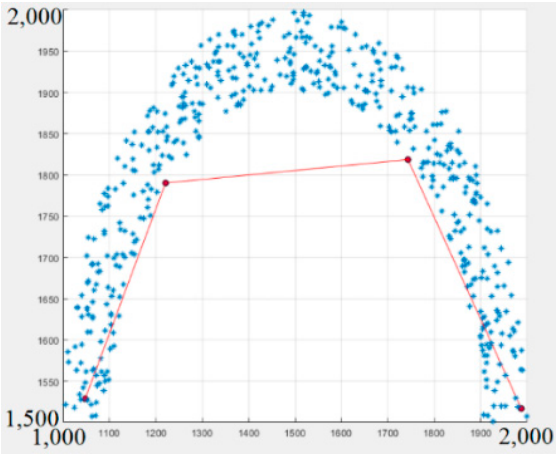
5.4. Restriction of Random Sampling

Most of the existing computer-vision-based skeleton detection algorithms use random sampling, as does our network skeleton detection algorithm. As a result, even with the same input of points cloud data, the skeleton extraction result may differ from each other, as shown in Figure 10b–d. This affects our routing protocol design. If many nodes run the skeleton detection algorithm simultaneously, it may cause some inconsistency. In our scheme, a master is selected to run the skeleton detection algorithm. Then, the master distributes the result to the entire network. Finally, all nodes have the same view of the

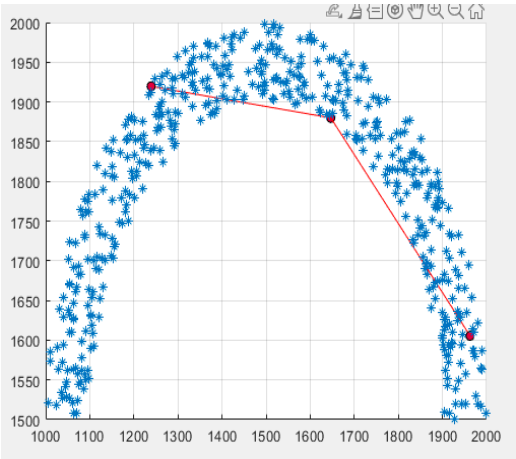
network skeleton. Section 6 will discuss SIP and how the masters distribute the skeleton information to the whole network.



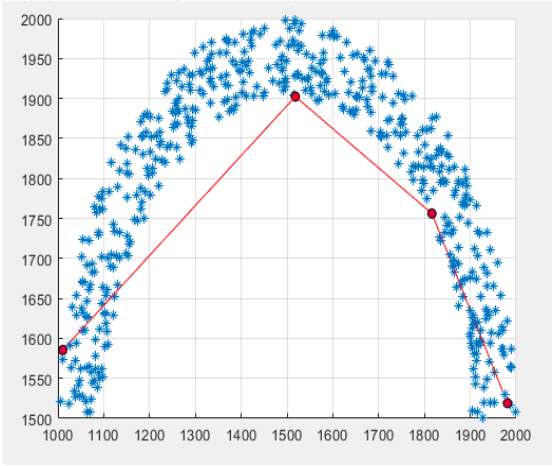
(**a**) skeleton for 100 m × 50 m (just one point)



(**b**) skeleton (solution 1) for 1000 m × 500 m



(**c**) skeleton (solution 2) for 1000 m × 500 m



(**d**) skeleton (solution 3) for 1000 m × 500 m

Figure 11. Detected skeletons for different cases.

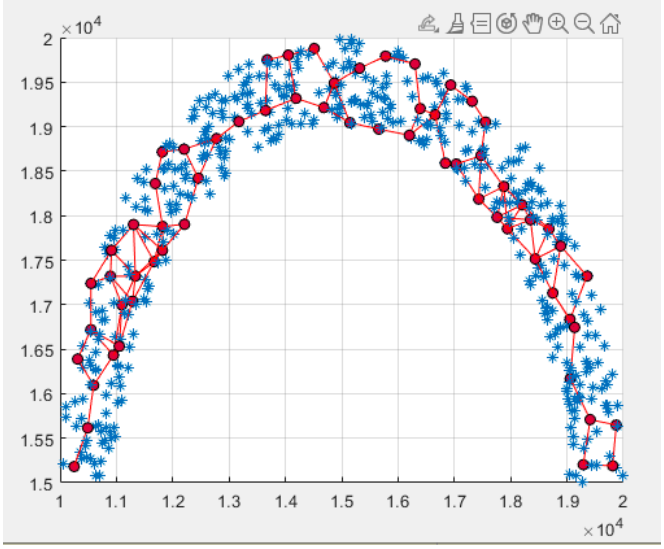


Figure 12. Detected skeleton for 10,000 m × 500 m swarm.

6. Skeleton Information Distribution and Routing Table Establishment

The selected masters have the network skeleton information for their local network regions. A set of skeleton information distribution policies was built to make sure every single node can keep tracking network topology changes. In particular, the skeleton information packets (SIPs) are designed to carry and distribute (1) network skeleton information, (2) nodes' position information, and (3) clustering information.

6.1. SIP Structure

The selected UAVs encapsulate all the skeleton information and other necessary information into an SIP and distribute it to the network. Figure 13 shows an example of an SIP generated by the master node 10.0.0.4. The packet contains seven parts. The first part is the packet header. The second part indicates the master node's IP address. The third and fourth parts contain skeleton information indices (already shown in Figure 5). The fifth part shows the connectivity information among skeleton nodes, and the sixth part is the position table. The seventh part records the time at which the master node distributes this packet.

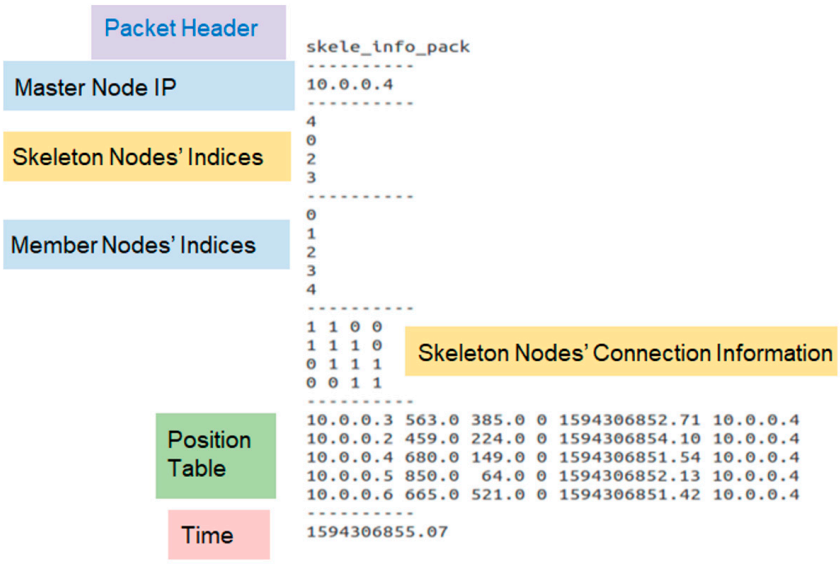


Figure 13. Format of SIP.

6.2. Improved Skeleton Information Distribution Scheme

If only one master is selected, each node simply forwards the SIP to all neighbors. However, for multiple masters, the distribution policy depends on the clustering results. Here, we use an example to illustrate the process. As mentioned before, each master distributes the SIP to its group members only. Therefore, whenever an SIP arrives, the node first checks the group member information in the packet. If the node is one of the members of the cluster, it forwards the SIP and updates the skeleton information saved in its local buffer.

In Figure 14, two nodes, 10.0.0.11 and 10.0.0.14, are selected as the master nodes, both highlighted as rectangle boxes. The master for cluster 1 is highlighted by a solid rectangle box, its group members are highlighted with solid cycles. Cluster 2 is highlighted with dash cycles and a rectangle box. The SIP eventually arrives at node 10.0.0.10, which examines the clustering information in the packets and finds out that it is a member of master 10.0.0.11. Then, only the SIP created by 10.0.0.11 is forwarded further.

However, the K-means and many other clustering algorithms cannot converge well within limited iterations. Figure 15 shows different K-means clustering results for the same swarm. In both tests, three clusters are formed within five iterations. The nodes inside the

rectangle boxes belong to different groups in two tests. Hence, the nodes at the edge may get confused about which group they belong to. We can call this situation the edge effect.

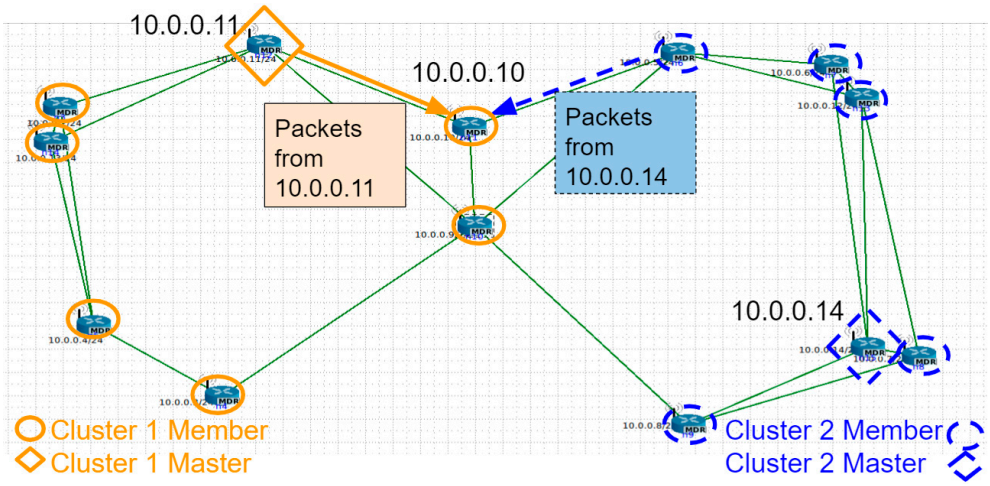
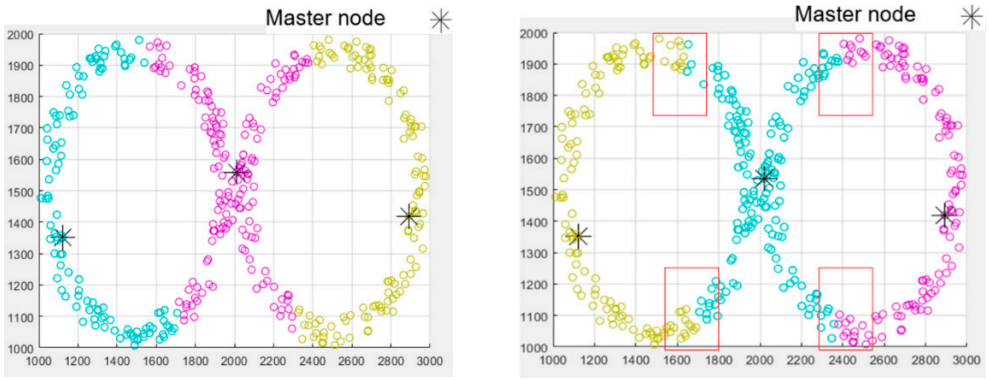


Figure 14. SIP distribution without edge conflict.



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

Figure 15. (**a**). K-means clustering result 1 (K = 3 with 5 iterations). (**b**) Result 2. Axes are in units of meters.

Figures 16 and 17 show two situations caused by the edge effect. In Figure 16, two masters claim that 10.0.0.10 is a member of their respective groups. Hence, both SIPs include 10.0.0.10 in the member node indices (part of the SIP). To address this issue, both packets will be forwarded. The skeleton detection algorithm may generate two sets of skeletons. Frequent changes of local skeleton information may cause 10.0.0.10 to update its routing path frequently. In Figure 17, none of the masters claims 10.0.0.10 as its member. A timeout value is used to address this issue. If a node's local skeleton information is outdated, the node still forwards the SIP.

6.3. Routing Table Creation and Modification

After the SIP is distributed to the entire network, each node holds the network skeleton information. In Stage III of the skeleton routing protocol, a conversion process is employed to transfer the skeleton information into a real routing table. Skeleton nodes and none-skeleton nodes use different policies to build up the routing table. The non-skeleton nodes look at the position table shown in Figure 14. The nearest skeleton node is the gateway to reach other IPs. A skeleton node examines the SI packet's skeleton node indices (i.e., skeleton node indices matrix SNIM) and connection information (i.e., skeleton connection representation matrix SCRM). A direct link is established between the local node and the

directly connected skeleton node. Other skeleton nodes can be added to the routing table by keeping track of the connection representation matrix SCRM.

Figure 18 is an expansion of Figure 6. It illustrates how to trace the matrix and find out the proper gateway. The local node (with an IP address of 10.0.0.5) can build a direct link with node 10.0.0.3. Hence, the first row adds destination (IP 10.0.0.3) with gateway 0.0.0.0. Through the analysis of the skeleton node 10.0.0.4 in the connection representation matrix SCRM, a connection can be established between 10.0.0.5 and 10.0.0.4 through the gateway 10.0.0.3. Similarly, 10.0.0.2 can establish a link with 10.0.0.5 through gateway 10.0.0.3. The non-skeleton node (or called 'leaf' nodes) can be added to the routing table after finding the closest skeleton node.

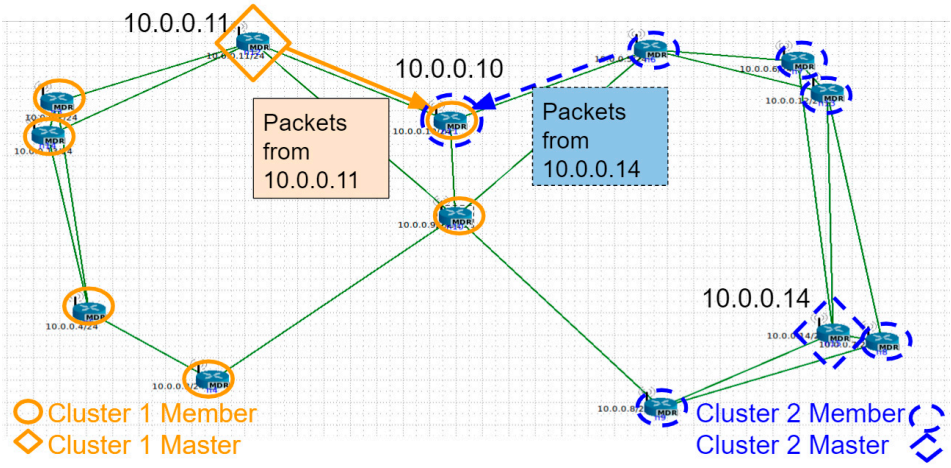


Figure 16. Multiple masters claim the same member.

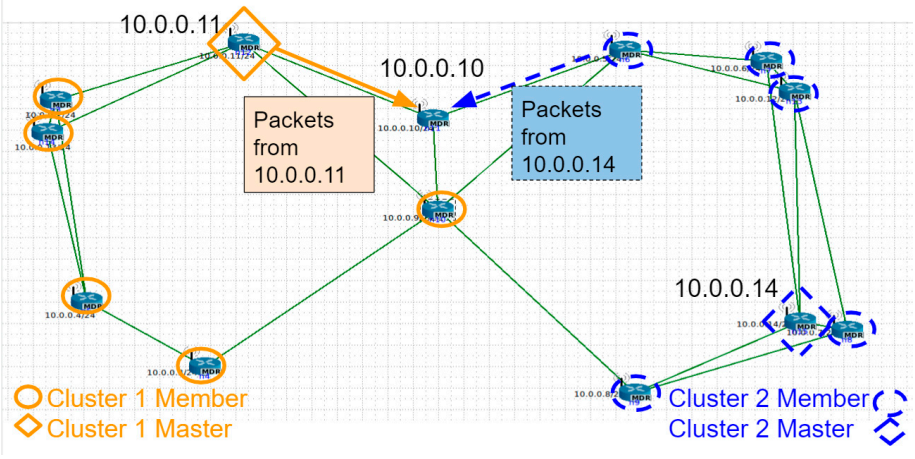


Figure 17. No master claims a specific node.

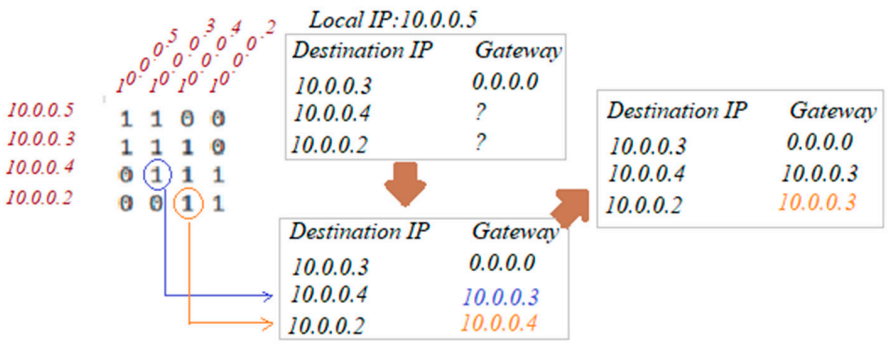


Figure 18. Adding skeleton node to the routing table by tracing the *skeleton connection matrix*.

7. Simulation and Tests Results

To test the performance of our skeleton routing scheme, CORE and EMANE emulation bundles are utilized. OLSR is used as the comparison baseline since it is one of the most popular proactive routing protocols used in an ad hoc network. Under the CORE simulation environment, EMANE acts as a plug-in and response for physical and link-layer modeling. WiFi 802.11b was chosen as the bottom layer protocol. Switching the MAC layer model in emulator will result in a different bit error rate at the same SINR. The QoS result may shift up and down a little. However, curve trends will not be affected. Table 1 shows MAC sublayer and antenna settings in our simulation. The concave size is 3000 m $\times$ 1000 m. A random walk model with constant speed is utilized through the tests. We define one specific node as the server that keeps sending 100 kbps UDP packets (1250 bytes for each packet) to all other nodes in the network. The master node number is kept as 1 to avoid potential edging effect issues.

Table 1. Simulation parameters (in physical and MAC layers).

Mode	802.11b (DSSS Only)
Enable promiscuous mode	off
Max distance (m)	1000
Unicast rate (Mbps)	11
Multicast rate (Mbps)	1
RTS threshold (bytes)	0
Enable traffic flow control	off
Number of flow control tokens	10
WiFi Multimedia (WMM)	off
Queue size (0:4: size)	0:255 1:255 2:255 3:255
Min contention window (0–4: minw)	0:32 1:32 2:16 3:8
Max contention window (0–4: maxw)	0:1024 1:1024 2:64 3:16
Arbitration inter frame space (0–4: aifs)	0:2 1:2 2:2 3:1
Txop (0–4: usec)	0:0 1:0 2:0 3:0
Retry limit (0–4: numertries)	0:3 1:3 2:3 3:3
Subid	1
System noise figure (dB)	4
Transmit power (dBm)	0
Antenna gain (dBi)	0
Noise processing mode	on
Noise bin size in microseconds	20
Path loss model	2-ray

7.1. Throughput, Packets Receive Rate, and Overhead vs. Swarm Size

We first tested the throughput, packet reception rate, and protocol overhead. A higher packet reception rate means a lower packet loss rate (thus higher link reliability). Each node moves around at a speed of 30 m/s. The swarms with sizes 10, 15, 20, 25, 25, and 30 were tested. Figures 19–21 show the test results for throughput, packet reception rate, and overhead, respectively. Our routing scheme outperformed OLSR in all those three metrics. As the system size increases, there will be a less chance for the network to split into a disconnected swarm. Hence, both skeleton routing and OLSR cases showed throughput increases at the beginning. However, our scheme can detect traffic paths with better stability and less chance to experience a path failure. Our scheme, in turn, provides

better throughput as well as packet delivery ratio. In addition, skeleton routing's protocol overhead ratio (measured by the percentage of protocol messages and overall packets sent) has a much lower increasing rate when the network (swarm) size is increasing. This is because our scheme only has compact position information (or link information if GPS is not available) and skeleton information needs to be flooded in the network. Even with limited wireless channel consumption, our protocol still provided traffic paths with better quality. Due to the lower overhead ratio, our routing scheme can also have more wireless channel resources available for the user's data.

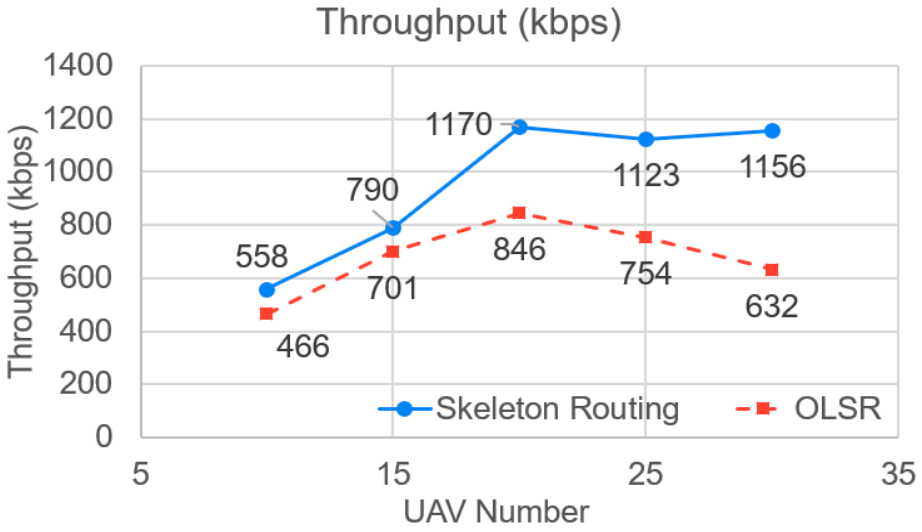


Figure 19. Throughput results with varying network sizes.

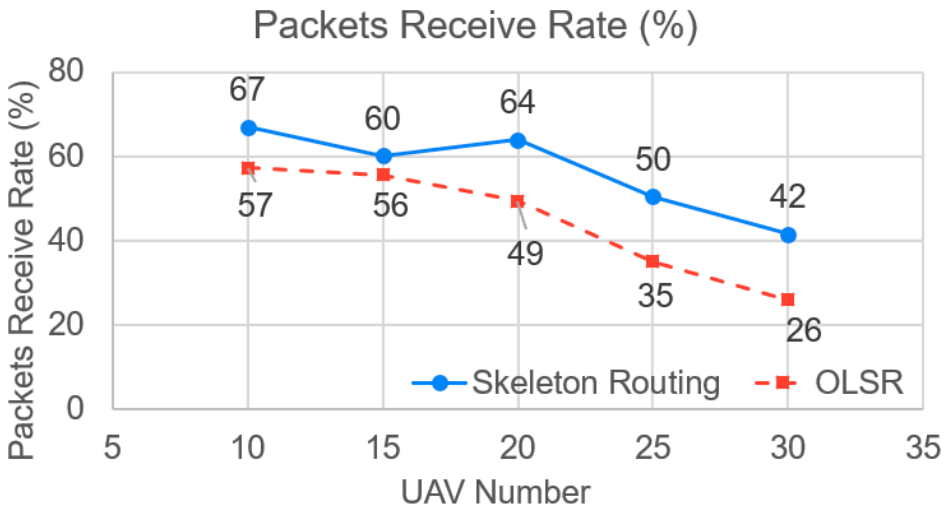


Figure 20. Packets reception rate results with varying network sizes.

7.2. Performance with Different Mobility Speeds

Further tests were performed to verify the effects of node movement speeds. The network size was kept at 20. Constant node speeds with 10, 20, 30, and 40 m/s were tested. Figure 22 shows the throughput results. Figure 23 shows packet reception rate results. Our routing scheme performed better than OLSR for all four-speed settings. The speed does not have big effects on protocol overhead. The OLSR has around 60% overhead for all situations, our routing scheme has around 13% overhead.

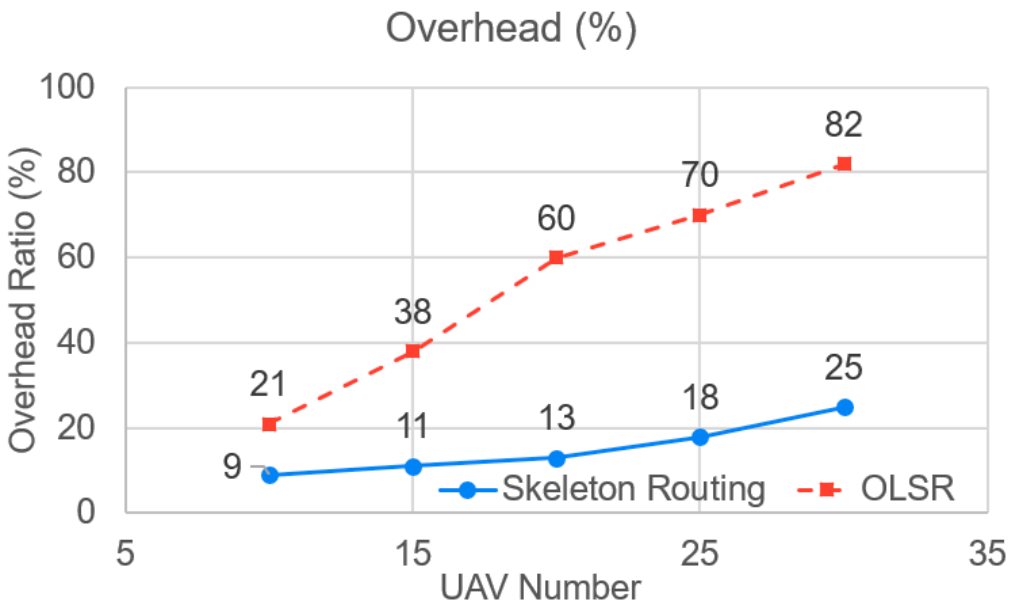


Figure 21. Overhead results with varying network sizes.

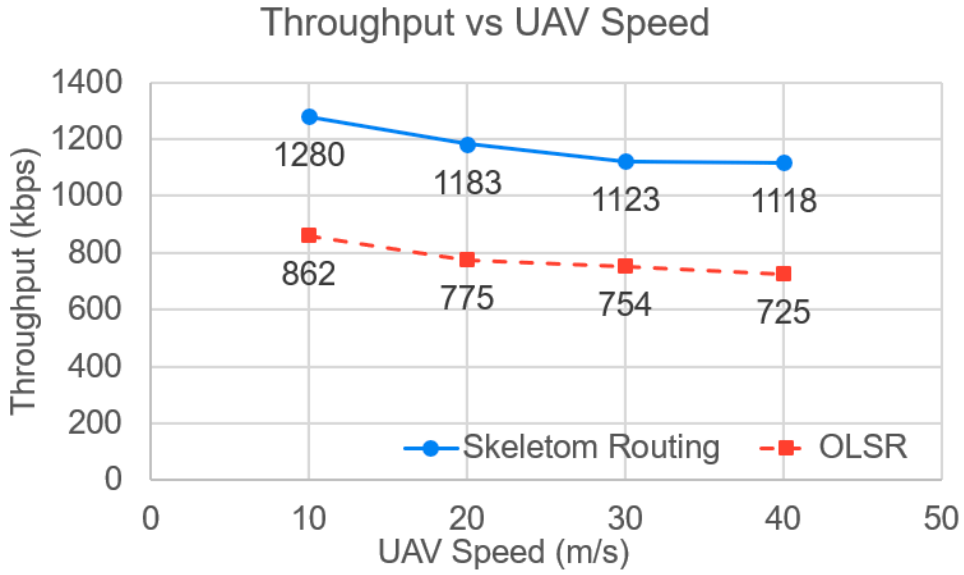


Figure 22. Throughput results with varying node speeds.

To comprehensively investigate the performance comparisons between our skeleton routing and OLSR scheme under multiple communication pairs (i.e., multiple sources send data to their destinations), we have measured the throughput changes under various network sizes and node speeds. The existence of multiple pairs reflects the actual scenarios in MANETs. Those traffic flows may cause wireless interference to each other if their path nodes are close to each other. As shown in Figure 24, in any mobility speeds and network sizes, our routing scheme has higher throughput performance. However, as the nodes number increases, the advantages are trend to decrease. That is because multiple traffic flows may share the same extract network frame as traffic path. This leads to a more detailed trunks and steams path selection algorithm to avoid medium contention, which left for future works.

Figure 25 shows the protocol overhead comparisons under different conditions. As we can see, when the network size gets larger, our protocol shows better overhead performance. This is important in large-scale networks that require a low protocol overhead.

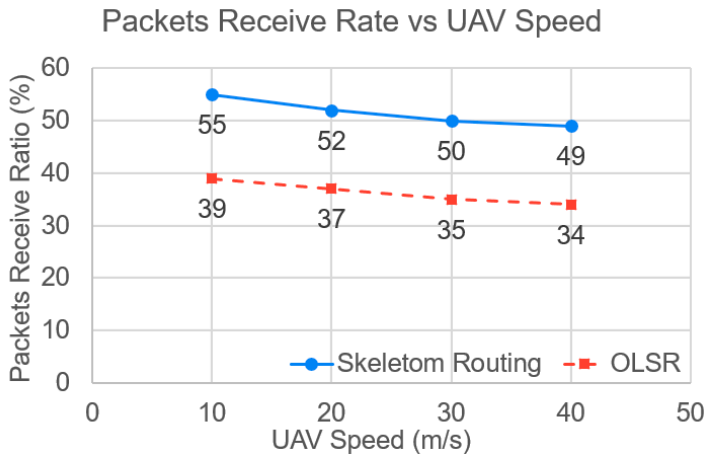


Figure 23. Packets reception rate results with varying speeds.

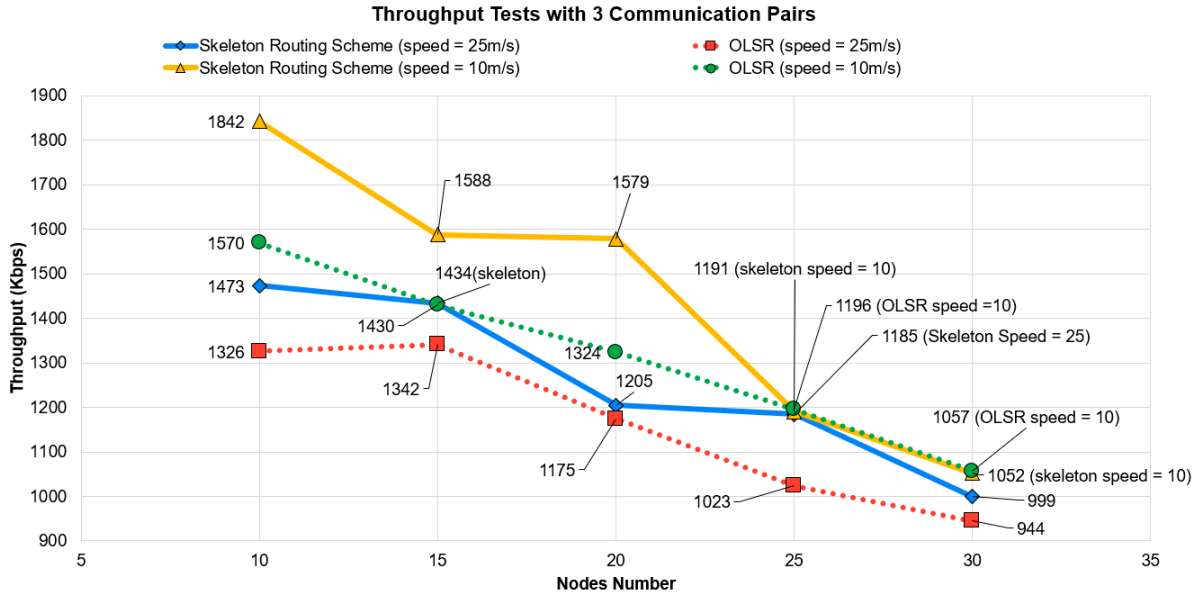


Figure 24. Throughput with different network sizes and node speeds.

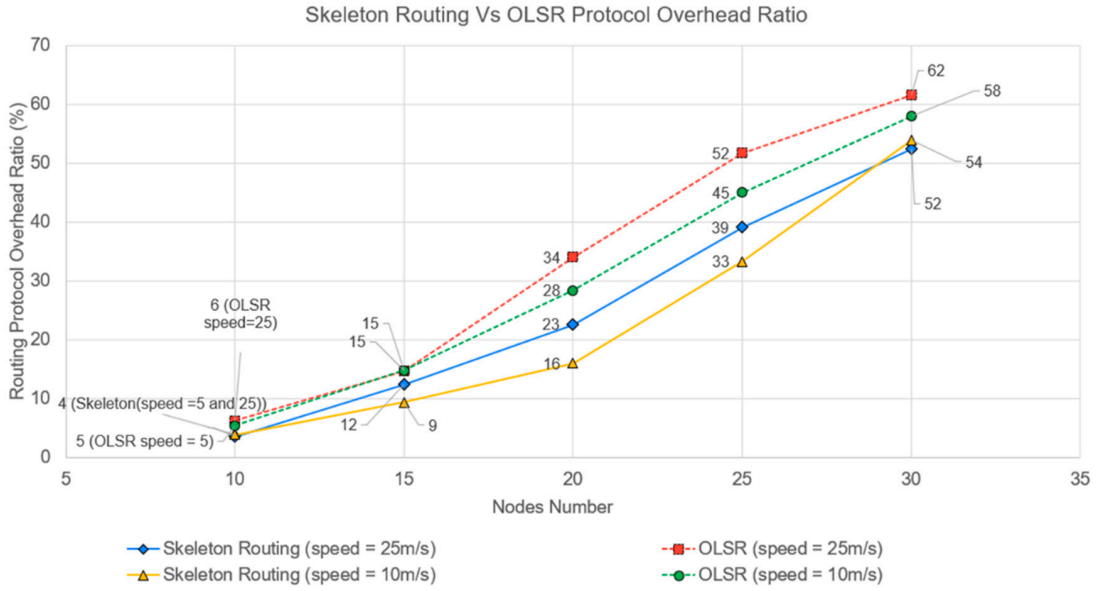


Figure 25. Protocol overhead with different network sizes and node speeds.

8. Conclusions

In this paper, we have presented a new routing scheme for MANETs that may change network topologies based on different missions. The T-UAV networks do not use random mobility styles since the tasks require all nodes to maintain a certain network shape. When the shapes change with task changes, the network always tries to minimize the node position changes to save node energy. Therefore, there often exist some relatively stable network regions during shape changes. In other words, the network has a certain framework during evolutional shape transformations. Thus, we designed a framework-based routing scheme that captures the main skeleton nodes that could bring a relatively stable routing path. Recapping that, our work contributes to three aspects: (1) Network framework extraction algorithm. We have provided an improved computer version algorithm with post-processing to extract such a skeleton—the extracted skeleton can perfectly serve stable traffic path establishment, making it easy for each node to update its routing table based on a recursive depth-first search method. (2) Detailed clustering protocol design and corresponding multi-cluster routing protocol operation schedule. A MANET clustering and message flooding management protocol was further introduced to help to raise cluster heads and eliminate potential unestablished distributed algorithm issues. (3) Realistic Unix system implementation and simulation. We have used the CORE+EMANE bundle tool to implement such a scheme into Unix. With detailed packet-level design, the simulation codes could be directly used in practical network products. The simulation results have verified the QoS advantage of our proposed scheme compared with a popular routing scheme—OLSR. However, we should still consider the limitations of our scheme. For example, we always assume nodes at the network core area are much more stable than edge nodes. However, networks with more freedom may have this assumption wrong, which brings difficulties for framework node detection. In addition, a medium contention issue might also be concerned. Multiple traffic flows may share the same extracted frame as the traffic path, which may cause congestion issues in extreme cases.

In future work, we will further investigate the 3D network skeleton extraction modeling and simulation schemes. Moreover, we will use a deep learning algorithm to recognize the movement patterns of task-oriented network nodes. CORE+EMANE will be used to compare our intelligent 3D skeleton routing scheme to other ones.

Author Contributions: Conceptualization, Z.C.; methodology; software; validation, Z.Y., J.Z., L.H. and I.R.; formal analysis, I.R.; investigation, Z.Y. and I.R.; resources; data curation, Z.Y., J.Z. and L.H.; writing—original draft preparation, Z.C.; writing—review and editing, Z.C., J.Z., L.H. and I.R.; visualization, Z.C.; supervision, I.R.; project administration, I.R.; funding acquisition, Not applicable. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Rubenstein, M.E.A. Kilobot: A Low Cost Robot with Scalable Operations Designed for Collective Behaviors. *Robot. Autonom. Syst.* **2014**, *62*, 966–975. [CrossRef]
2. Anderson, B.D.; Fidan, B.; Yu, C.; Walle, D. UAV Formation Control: Theory and Application. In *Recent Advances in Learning and Control*; Springer: London, UK, 2008; pp. 15–33.
3. Yong, E. Autonomous Drones Flock Like Birds: Copters can Arrange in Formation and Coordinate Flight Patterns without Central Control. *Nature* **2014**, *26*, 61244441.
4. Drone Light Show in China. Available online: https://dronedj.com/2019/11/20/drone-light-shows-china/ (accessed on 20 December 2021).

5. Cho, S.; Park, H. OLSR Protocol Having a Link Reliability List in the MPR. In Proceedings of the 12th International Conference on Advanced Communication Technology (ICACT), Gangwon, Korea, 7–10 February 2010; pp. 1231–1234.
6. Lochert, C.; Scheuermann, B.; Mauve, M. A Survey on Congestion Control for Mobile Ad Hoc Networks. *Wirel. Commun. Mob. Comput.* **2007**, *75*, 655–676. [CrossRef]
7. Wang, F.; Chen, Z.; Zhang, J.; Zhou, C.; Yue, W. Greedy Forwarding and Limited Flooding based Routing Protocol for UAV Flying Ad-Hoc networks. In Proceedings of the IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 12–14 July 2019; pp. 1–4. [CrossRef]
8. Lin, L.; Sun, Q.; Wang, S.; Yang, F. A Geographic Mobility Prediction Routing Protocol for Ad Hoc UAV Network. In Proceedings of the 2012 IEEE Globecom Workshops, Anaheim, CA, USA, 3–7 December 2012; pp. 1597–1602. [CrossRef]
9. Dong, S.Y. Optimization of OLSR Routing Protocol in UAV Ad HOC Network. In Proceedings of the 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 16–18 December 2016; pp. 90–94. [CrossRef]
10. Mohapatra, S.; Tripathy, T. MM-OLSR: Multi Metric Based Optimized Link State Routing Protocol for Wireless Ad-Hoc Network. In Proceedings of the 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, 3–5 October 2016; pp. 153–158. [CrossRef]
11. Solanki, Z.G.; Mahida, P.T. SR-AODV: Modified AODV to Avoid Link Breakage in Wireless Mesh Network. In Proceedings of the 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 6–8 April 2016; pp. 181–186. [CrossRef]
12. Qi, G.; Ying, L.; Jie, L. An Optimized Adhoc Network Routing Protocol for Multi-Performance Decision Making. In Proceedings of the IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 14–16 December 2018; pp. 1946–1949. [CrossRef]
13. Chatterjee, M.; Das, S.K.; Turgut, D. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. *Clust. Comput.* **2002**, *5*, 193–204. [CrossRef]
14. Fotohi, R.; Ebazadeh, Y.; Geshlag, M.S. A New Approach for Improvement Security against DoS Attacks in Vehicular Ad-Hoc Network. *arXiv* **2020**, arXiv:2002.10333. [CrossRef]
15. Zhang, D.; Ge, H.; Zhang, T.; Cui, Y.Y.; Liu, X.; Mao, G. New Multi-Hop Clustering Algorithm for Vehicular Ad Hoc Networks. *IEEE Trans. Intell. Transport. Syst.* **2018**, *20*, 1517–1530. [CrossRef]
16. Huang, H.; Wu, S.; Cohen-Or, D.; Gong, M.; Zhang, H.; Li, G.; Chen, B. L1-Medial Skeleton of Point Cloud. *ACM Trans. Graph.* **2013**, *32*, 1–8. [CrossRef]